Advant® OCS
with Master software

AdvaInform® Object Types

Reference Manual

ABB

Use of **DANGER**, **WARNING**, **CAUTION**, and **NOTE**

This publication includes, **DANGER**, **WARNING**, **CAUTION**, and **NOTE** information where appropriate to point out safety related or other important information.

DANGER	Hazards which could result in severe personal injury or death
WARNING	Hazards which could result in personal injury
CAUTION	Hazards which could result in equipment or property damage
NOTE	Alerts user to pertinent facts and conditions

Although **DANGER** and **WARNING** hazards are related to personal injury, and **CAUTION** hazards are associated with equipment or property damage, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all **DANGER**, **WARNING**, and **CAUTION** notices.

TRADEMARKS

Advant, AdvaCommand, AdvaInform, AdvaBuild, AdvaControl, and ABB Master are registered trademarks of ABB Asea Brown Boveri Ltd. Switzerland

DEC, DECNet, Digital, VAX and VMS are trademarks of Digital Equipment Corporation

HP and all HP-products mentioned in this publication are registered trademarks of Hewlett Packard Company

MODBUS is a registered trademark of GOULD Electronics

ORACLE, CASE*Dictionary, Easy*SQL, SQL*Calc, SQL*Connect, SQL*Design Dictionary, Oracle Forms, SQL*Loader, SQL*Menu, SQL*Net, SQL*Plus, SQL*Report, Pro*Ada, Pro*Cobol, Pro*Fortran, ProPascal, Pro*C, Oracle*Mail, Oracle*Terminal, CASE*Designer, CASE*Generator, PL/SQL, SQL*DBA, and SQL*Reportwriter are trademarks of Oracle Corporation

UNIX is a registered trademark of AT & T Corporation

NOTICE

The information in this document is subject to change without notice and should not be construed as a commitment by ABB Industrial Systems AB. ABB Industrial Systems AB assumes no responsibility for any errors that may appear in this document.

In no event shall ABB Industrial Systems AB be liable for direct, indirect, special, incidental or consequential damages of any nature or kind arising from the use of this document, nor shall ABB Industrial Systems AB be liable for incidental or consequential damages arising from use of any software or hardware described in this document.

This document and parts thereof must not be reproduced or copied without ABB Industrial Systems AB’s written permission, and the contents thereof must not be imparted to a third party nor be used for any unauthorized purpose.

The software described in this document is furnished under a license and may be used, copied, or disclosed only in accordance with the terms of such license.

CE MARKING

This product meets the requirements specified in EMC Directive 89/336/EEC and in Low Voltage Directive 72/23/EEC.

Copyright © ABB Industrial Systems AB 1997.

TABLE OF CONTENTS

Chapter 1 - Introduction

1.1 General Information 1-1
1.2 Manual Organization 1-1
1.3 Conventions 1-1
1.4 Related Documentation 1-2
1.5 Terminology 1-2
1.6 Data Sheet Content 1-3
  1.6.1 General 1-3
  1.6.2 Graphical Overview 1-4
  1.6.3 Attribute Section 1-5
  1.6.4 Connection Section 1-6
  1.6.5 Composite Attribute Section 1-6
  1.6.6 Command Section 1-6
    1.6.6.1 General 1-6
    1.6.6.2 Commands Available for IMS User Objects 1-7
    1.6.6.3 Commands Available for the Majority of ABB Master Objects 1-8
    1.6.6.4 Examples 1-9
  1.6.7 Event Section 1-13
    1.6.7.1 General 1-13
    1.6.7.2 Events Available for the Majority of ABB Master Objects 1-13
1.7 Messages to Events and Alarm Subscribers 1-14
1.8 Error Messages 1-14
1.9 How to Use the Data Sheets 1-14
  1.9.1 C/C++ Declarations Needed by Object Access Services 1-14
  1.9.2 Access via SQL 1-16
    1.9.2.1 IMS_USER 1-16
    1.9.2.2 Example on Access to a Process Object 1-17
  1.9.3 Access from User Objects and Calculations 1-17
1.10 Restrictions 1-17

Chapter 2 - Inheritance

2.1 General 2-1
2.2 Basic Inheritance 2-1

CONTENTS (continued)

Chapter 3 - Process Objects

3.1 Analog Input - AI 3-1
  3.1.1 How to Use 3-1
  3.1.2 Overview 3-3
  3.1.3 Attributes 3-4
  3.1.4 Connections 3-6
  3.1.5 Composite Attributes 3-6
  3.1.6 Commands 3-7
  3.1.7 Events 3-8
  3.1.8 SQL Views 3-9
3.2 Analog Output - AO 3-12
  3.2.1 How to Use 3-12
  3.2.2 Overview 3-14
  3.2.3 Attributes 3-15
  3.2.4 Connections 3-16
  3.2.5 Composite Attributes 3-17
  3.2.6 Commands 3-17
  3.2.7 Events 3-18
  3.2.8 SQL Views 3-19
3.3 Digital Input - DI 3-21
  3.3.1 How to Use 3-21
  3.3.2 Overview 3-22
  3.3.3 Attributes 3-23
  3.3.4 Connections 3-25
  3.3.5 Composite Attributes 3-25
  3.3.6 Commands 3-26
  3.3.7 Events 3-27
  3.3.8 SQL Views 3-27
3.4 Digital Output - DO 3-29
  3.4.1 How to Use 3-29
  3.4.2 Overview 3-31
  3.4.3 Attributes 3-32
  3.4.4 Connections 3-33
  3.4.5 Composite Attributes 3-34
  3.4.6 Commands 3-34
  3.4.7 Events 3-35
  3.4.8 SQL Views 3-35

CONTENTS (continued)

3.5 DAT Object - DAT 3-38
  3.5.1 How to Use 3-38
  3.5.2 Overview 3-39
  3.5.3 Attributes 3-40
  3.5.4 Connections 3-40
  3.5.5 Composite Attributes 3-40
  3.5.6 Commands 3-41
  3.5.7 Events 3-42
  3.5.8 SQL Views 3-42
3.6 TEXT Object - TEXT 3-44
  3.6.1 How to Use 3-44
  3.6.2 Overview 3-45
  3.6.3 Attributes 3-46
  3.6.4 Connections 3-47
  3.6.5 Composite Attributes 3-47
  3.6.6 Commands 3-47
  3.6.7 Events 3-49
  3.6.8 SQL Views 3-50
3.7 Process Controller - PIDCON 3-52
  3.7.1 How to Use 3-52
  3.7.2 Overview 3-53
  3.7.3 Attributes 3-54
  3.7.4 Connections 3-59
  3.7.5 Composite Attributes 3-59
  3.7.6 Commands 3-60
  3.7.7 Events 3-63
  3.7.8 SQL Views 3-63
3.8 Ratio Station - RATIOSTN 3-71
  3.8.1 How to Use 3-71
  3.8.2 Overview 3-72
  3.8.3 Attributes 3-73
  3.8.4 Connections 3-76
  3.8.5 Composite Attributes 3-77
  3.8.6 Commands 3-78
  3.8.7 Events 3-80
  3.8.8 SQL Views 3-81

CONTENTS (continued)

3.9 Manual Station - MANSTN 3-88
    3.9.1 How to Use 3-88
    3.9.2 Overview 3-89
    3.9.3 Attributes 3-90
    3.9.4 Connections 3-93
    3.9.5 Composite Attributes 3-93
    3.9.6 Commands 3-94
    3.9.7 Events 3-96
    3.9.8 SQL Views 3-97
3.10 General Binary Object - GENBIN 3-104
    3.10.1 How to Use 3-104
    3.10.2 Overview 3-105
    3.10.3 Attributes 3-105
    3.10.4 Connections 3-108
    3.10.5 Composite Attributes 3-109
    3.10.6 Commands 3-109
    3.10.7 Events 3-111
3.11 General Controller - GENCON 3-112
    3.11.1 How to Use 3-112
    3.11.2 Overview 3-113
    3.11.3 Attributes 3-114
    3.11.4 Connections 3-116
    3.11.5 Composite Attributes 3-117
    3.11.6 Commands 3-117
    3.11.7 Events 3-120
3.12 General User Defined Object - GENUSD 3-121
    3.12.1 How to Use 3-121
    3.12.2 Overview 3-122
    3.12.3 Attributes 3-123
    3.12.4 Connections 3-125
    3.12.5 Composite Attributes 3-126
    3.12.6 Commands 3-126
    3.12.7 Events 3-130

CONTENTS (continued)

3.13 Valve Control - VALVECON 3-131
  3.13.1 How to Use 3-131
  3.13.2 Overview 3-132
  3.13.3 Attributes 3-133
  3.13.4 Connections 3-143
  3.13.5 Composite Attributes 3-143
  3.13.6 Commands 3-144
  3.13.7 Events 3-145
3.14 Motor Control - MOTCON 3-145
  3.14.1 How to Use 3-145
  3.14.2 Overview 3-147
  3.14.3 Attributes 3-147
  3.14.4 Connections 3-153
  3.14.5 Composite Attributes 3-153
  3.14.6 Commands 3-154
  3.14.7 Events 3-157
3.15 Uninterruptable Power Supply - UPS 3-158
  3.15.1 How to Use 3-158
  3.15.2 Overview 3-160
  3.15.3 Attributes 3-160
  3.15.4 Connections 3-162
  3.15.5 Composite Attributes 3-163
  3.15.6 Commands 3-163
  3.15.7 Events 3-164
3.16 coExecutor Object 3-165
  3.16.1 How to Use 3-165
  3.16.2 Overview 3-165
  3.16.3 Attributes 3-166
  3.16.4 Connection 3-167
  3.16.5 Composite Attribute 3-167
  3.16.6 Commands 3-167
  3.16.7 Event 3-168

CONTENTS (continued)

3.17 coBasicCalc Object ... 3-169
  3.17.1 How to Use ... 3-169
  3.17.2 Overview ... 3-170
  3.17.3 Attributes ... 3-171
  3.17.4 Connections ... 3-171
  3.17.5 Composite Attributes ... 3-172
  3.17.6 Commands ... 3-172
  3.17.7 Normal Operations ... 3-172
  3.17.8 Event ... 3-172
3.18 coRunTimeMeasurement Object ... 3-172
  3.18.1 How to Use ... 3-172
  3.18.2 Overview ... 3-174
  3.18.3 Attributes ... 3-175
  3.18.4 Connection ... 3-176
  3.18.5 Commands ... 3-176
  3.18.6 Events ... 3-177

Chapter 4 - System Application Objects

4.1 Object Handler Supervisor Object - Daemon ... 4-1
  4.1.1 How to Use ... 4-1
  4.1.2 Overview ... 4-2
  4.1.3 Attributes ... 4-3
  4.1.4 Connections ... 4-3
  4.1.5 Composite Attributes ... 4-3
  4.1.6 Commands ... 4-3
  4.1.7 Events ... 4-4
4.2 ObjectHandler Object - ObjectHandlerType ... 4-5
  4.2.1 How to Use ... 4-5
  4.2.2 Overview ... 4-5
  4.2.3 Attributes ... 4-6
  4.2.4 Connections ... 4-6
  4.2.5 Composite Attributes ... 4-6
  4.2.6 Commands ... 4-6
  4.2.7 Events ... 4-7
  4.2.8 Object Handler Names ... 4-7

CONTENTS (continued)

4.3 Event & Alarm Distributor Object - EADISTR 4-8
  4.3.1 How to Use 4-8
  4.3.2 Overview.. 4-8
  4.3.3 Attributes 4-8
  4.3.4 Connections 4-9
  4.3.5 Composite Attributes 4-9
  4.3.6 Commands 4-9
  4.3.7 Events.. 4-13
  4.3.8 EADISTR Names 4-13
4.4 Calculation Object - CalcObject 4-14
  4.4.1 How to Use 4-14
  4.4.2 Overview.. 4-15
  4.4.3 Attributes 4-16
  4.4.4 Connections 4-17
  4.4.5 Composite Attributes 4-17
  4.4.6 Commands 4-18
  4.4.7 Events.. 4-18
  4.4.8 CalcObject Names 4-18
4.5 Report Object -ReportObject 4-19
  4.5.1 How to Use 4-19
  4.5.2 Overview.. 4-19
  4.5.3 Attributes 4-20
  4.5.4 Connections 4-20
  4.5.5 Composite Attributes 4-21
  4.5.6 Commands 4-21
  4.5.7 Events.. 4-21
  4.5.8 ReportObject Names.. 4-21
4.6 Dataset Object - 4-22
  4.6.1 How to Use 4-22
  4.6.2 Overview.. 4-23
  4.6.3 Attributes 4-24
  4.6.4 Connections 4-25
  4.6.5 Operations.. 4-25
  4.6.6 Events.. 4-25
  4.6.7 User API program examples... 4-26

CONTENTS (continued)

4.7 System Status Object - ASNode 4-27
- 4.7.1 How to Use 4-27
- 4.7.2 Overview 4-27
- 4.7.3 Attributes 4-28
- 4.7.4 Connections 4-36
- 4.7.5 Composite Attributes 4-36
- 4.7.6 Commands 4-36
- 4.7.7 Events 4-36
- 4.7.8 System Status Names 4-36

4.8 Log Manager Object - LogMgr 4-37
- 4.8.1 How to Use 4-37
- 4.8.2 Overview 4-37
- 4.8.3 Attributes 4-38
- 4.8.4 Commands 4-42
- 4.8.5 Events 4-47
- 4.8.6 Log Manager Names 4-47

4.9 Numeric Log Object - NLog 4-48
- 4.9.1 How to Use 4-48
- 4.9.2 Overview 4-49
- 4.9.3 Attributes 4-50
- 4.9.4 Commands 4-52
- 4.9.5 Numeric Log Names 4-60
- 4.9.6 SQL Views 4-61

4.10 Message Log Object - MLog 4-63
- 4.10.1 How to Use 4-63
- 4.10.2 Overview 4-64
- 4.10.3 Attributes 4-65
- 4.10.4 Commands 4-67
- 4.10.5 Message Log Names 4-74

4.11 RTA System Object 4-75
- 4.11.1 How to Use 4-75
- 4.11.2 Overview 4-75
- 4.11.3 Attributes 4-76
- 4.11.4 Connections 4-76
- 4.11.5 Composite Attributes 4-76
- 4.11.6 Commands 4-76
- 4.11.7 Event 4-76

Chapter 1 Introduction

1.1 General Information

The application developer using Advant Station IMS as platform needs to have a good understanding of designing object oriented applications and programs, since all data and functions are handled as objects in the Advant OCS. For more information on object handling in the Advant OCS refer to the *AdvaInform Object Handling User's Guide* and *AdvaBuild Object Type Builder User's Guide.*

This document is mainly intended for a user who develops applications with AdvaBuild Object Type Builder or with Object Access Services in the AdvaInform User API. The description of attributes for the different Object Types is also of interest when using SQL access to the different objects.

Each description of the different object types includes a figure with an overview of the attributes, connections, commands, events and error messages that are valid for the specific object type. All the accessible attributes, available commands and events are then described in detail in separate sections. Available composite attributes for the object type are also described in detail.

There are two main categories of objects. Process objects and system application objects. Process objects are entities representing control and supervision functions. E.g: AI, AO, PIDCON. The system application objects are such as system status, log manager, numerical log, message log and report object.

1.2 Manual Organization

Chapter 1 and 2

Chapter 1 contains an introduction to the programming with AdvaInform User API and User Objects. Chapter 2 describes the ability to inherit.

Chapter 3 and 4 - Data sheets

The rest of the document contains data sheets. One for each documented object type. Chapter 3 contains Process Objects and Chapter 4 System Application Objects.

1.3 Conventions

The following typographic conventions are used in this document:

- The names of keyboard keys, push buttons, menus, and menu items are boldfaced. System prompts/messages are in the Courier font, and user responses/input are **bold**.
- The names of referenced manuals are *itali*cs

1.4 Related Documentation

The documentation related to this manual is shown in Figure 1-1

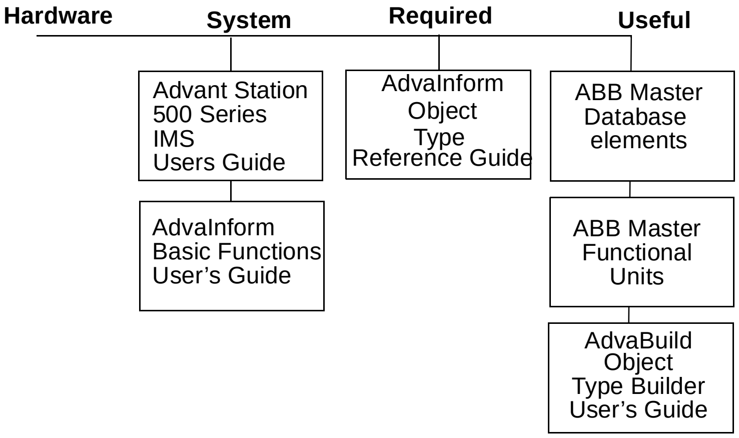

Hardware
System
Required
Useful
Advant Station 500 Series IMS Users Guide
AdvaInform Object Type Reference Guide
ABB Master Database elements
AdvaInform Basic Functions User's Guide
ABB Master Functional Units
AdvaBuild Object Type Builder User's Guide

Figure 1-1. AdvaInform Object Type Reference Guide and Related Documentation Tree

1.5 Terminology

OTDL: Object Type Definition Language. A description language making it possible to define all the different parts of an object type. All objects in the Advant OCS are described in OTDL.

Attribute: Describes the public data of the object. The result of an application is most commonly stored in attributes. Attributes also contain configured public information. Applications (other objects) can request the object's public data through the core system, using subscriptions.

Composite Attribute: A Composite Attribute is a list of attributes provided in one request. The receiver requests the public data using the core system subscription service and referring to the Composite Attribute that the other objects have declared.

Command: An Operation makes requests of an object. As a result, the object performs the requested action.

Connection: Note that Connections are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

A connection is a method to relate to another object. Data can be requested from or sent to other objects by the object itself. All interaction goes through the core system. Data is read from attributes of other objects. There are two types of input connections: Single and Object. A Single connections is a connection to any attribute of any object type of a certain data type. Which object instance and which attribute is decided upon at configuration. An object connection is a connection to a specified object type. Only objects of that object type may be configured to that connection. Output connections are always of Object Connection type. Output to other objects is produced by giving commands to the objects.

Core system: The basic software that provides services for communication between all applications/objects in the Advant OCS. Applications becomes fully distributed within Advant OCS. The core system is part of all products.

1.6 Data Sheet Content

1.6.1 General

The data sheets consist of the following parts:

- A graphical overview and a short description of the Object Type
- An attribute section
- A connection section
- A composite attribute section
- A command section
- An event section

1.6.2 Graphical Overview

Example: DI Functionality

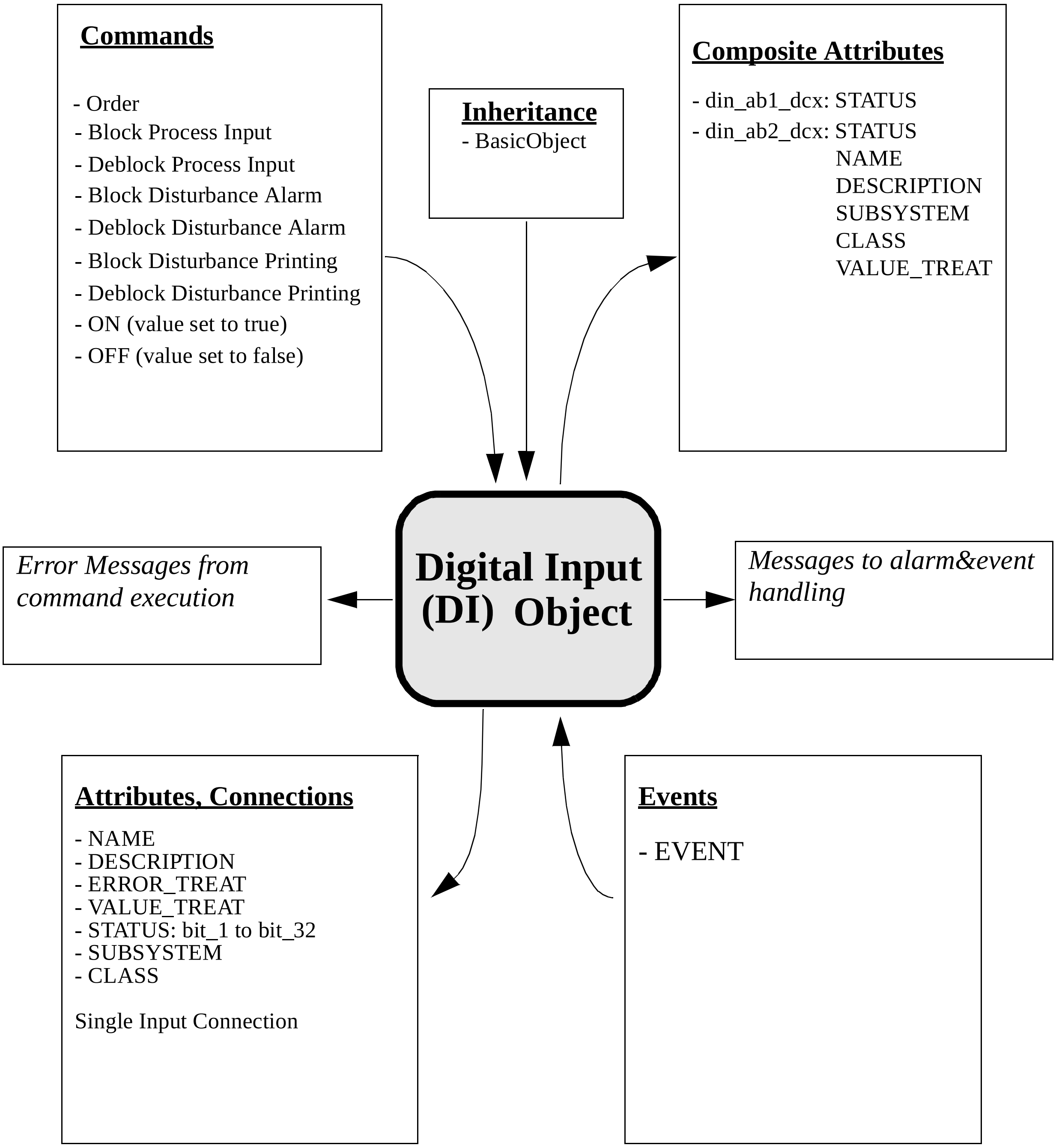


Figure 1-2. Overview of the Object Type DI.

The Attributes, Connections, Composite Attributes, Commands and Events are described in the sections that follow in the data sheet.

Error messages and event messages to the Event and Alarm system are not of interest for the application development. Thus, they are not described in the data sheets. See however Section 1.7, Messages to Events and Alarm Subscribers.

Inheritance shows the inheritance chain for the object type. Inheritance is only applicable for Object Types implemented as User Objects.

1.6.3 Attribute Section

For each Object Type a table describes the available attributes:

Table 1-1. Example of an attribute table for an object type.

Attribute	**OTDL Data Type**	**Description**
NAME	TdName (c)	The name of the object.
DESCRIPTION	TdDescription (c)	The description of the object.
ERROR_TREAT	IntShort (c)	The event treatment number in the Operator Station/MasterView. Determines how the error status shall be processed and presented in the event/alarm lists. If set to 0, no event will be generated.

The first column represents the attribute name.

NOTE

The attribute name is case sensitive.

The second column, Data Type, is the data type of the attribute as defined in OTDL (Object Type Definition Language). The note (c) shows if the attribute is configurable or not in Object Configuration display.

The third column is a description of the attribute.

When an attribute is collected the data is returned in a C-type named Omf_’ObjectTypeName’ + ‘AttributeName’.

1.6.4 Connection Section

The Connection description is only applicable on Object Types that are implemented as User Objects.

For each object type a table describes the connections to other objects. The table has the following format and content:

Table 1-2. Example on table describing connections

Name	Direction	Type/Format	Description
InputValue1	Input	Single / Float	Single floating point connection for first input parameter
InputValue2	Input	Single / Float	Single floating point connection for second input parameter
ResultAO	Output	Object (AO)	Output AO object connection for output of result

The Name column describes the name of the connection, the direction is input or output and the type/format describes the connection type (single or object) and if single the data type format. The description finally describes the purpose/usage of the connection.

1.6.5 Composite Attribute Section

When data is subscribed upon from an object the Composite Attribute always refers to one or several attributes. To simplify the subscriptions a number of predefined attribute lists are defined. These lists are called Composite Attributes. They are referred to just as if they where single attributes. The data is returned in a C struct described with the struct type definition named Omf_+’ObjectTypeName’+’CompositeAttributeName’. The name syntax of the attribute id is described in Section 1.9.1, C/C++ Declarations Needed by Object Access Services.

The Composite Attribute section presents the list of attributes included in each subscription. Composite Attributes are defined lists of several attributes which commonly are collected together to simplify collection.

1.6.6 Command Section

1.6.6.1 General

All commands to an object are described as follows:

- Syntax for each command. The name of the command and the order of the parameters.
- A table for each command describing the command parameters in detail.
- Complementary information to certain command parameters

Examples on how commands are described are given in Section 1.6.6.2, Commands Available for IMS User Objects and Section 1.6.6.3, Commands Available for the Majority of ABB Master Objects.

1.6.6.2 Commands Available for IMS User Objects

The following commands are available for all object types which are developed with the AdvaBuild Object Type Builder:

Table 1-3. Global commands available for all IMS User Objects

Command	Description
Delete	Delete stops the execution of the object, see Deactivate below, and then deletes the object. All primary and disk memory resources that the object allocated are released.
Deactivate	Deactivate stops the execution of the object. Its primary memory allocation is removed and it is no longer available in the run time environment.
Copy	Copy creates a copy of the object in question and gives it the given name. All other attributes are identical with the object from which it is copied.
NormalOperation	Execute the objects Normal Operation.

Syntax

- Delete ()
- Deactivate ()
- Copy (Name)
- NormalOperation ()

Parameters

The command Copy has the following parameter:

Table 1-4. Parameters to the Command Copy

Parameter	Direction	OTDL- and C-Format	Value/explanation
Name	input	TdName	The name of the new object instance to be created.

1.6.6.3 Commands Available for the Majority of ABB Master Objects

The ABB Master objects support the command ORDER described below. To rationalize, the ORDER command is described in this section. Each data sheet is then just complemented with detailed information about the specific functionalities provided for the object type.

Syntax

The syntax of the ORDER command is as follows:

ORDER (opCode, opProp, typeOfReq, eventLog, dialogID, aValue, dValue, status, textIndex)

Parameters

There are two input parameters that control the functionality of the ORDER command. The parameter opCode describes "what is to be done" and the parameter opProp "on what it is to be done". Example: The opCode tells the object to block a functionality and the opProp describes what functionality to block.

Table 1-5. Parameters to the Command ORDER

Parameter	Direction	OTDL- and C-Format	Value/explanation
opCode	input	IntSmall - OmfSmall	Type of order(1)
opProp	input	IntSmall - OmfSmall	Property involved(1)
eventLog	input	Boolean - OmfBoolean	True if the command shall be logged, else false. True generates an event message to the event & alarm system.
typeOfReq	input	IntSmall - OmfSmall	Program (dcPROG=1) or Operator (dcOPERATOR=0)(2)
dialogId	input	IntLong- OmfLong	0 for program
aValue	input	Float - OmfFloat	Analog value(3)
dValue	input	IntLong- OmfLong	Digital value(3)
status	output	IntSmall - OmfSmall	The operation result.(4) dcGRANT = 0 OK dcNOT_ALLOWED = 2 for failure.
textIndex	output	IntShort - OmfShort	Text index for operator message.

(1) See corresponding table with the parameters for the specific Object Type.
(2) typeOfReq = dcProg shall be used with Object Access Services
(3) One of the aValue and dValue parameters is often not required or used for an object type. E.g: to an AI the dValue parameter is not used.
(4) If there is an error in the input parameters, e.g. a value is outside a range or the request requires that the object is in a specific mode, an error message is sent back to the requestor. Such errors are reported in the two output parameters, status and textIndex, which are common to all commands. A status not equal to dcGRANT (0) upon return this indicates that an error has occurred and that textIndex has the value of the precise nature of the error. The corresponding error text to the value can be retrieved with AdvaInform User API.

Include files

The available opCode and opProp values and their combinations are described in detail in the respective data sheet. To include the constant declarations of the opCode, opProp, typeOfReq, status and textIndex parameters from the User API, use the include file:

`/opt/advant/UserAPI/include/dccodes.hh.`

1.6.6.4 Examples

The following is an example on how the command ORDER is executed from the AdvaInform User API using an Object Access service:

```
/*
// FILE         SetAO.c
//
// DESCRIPTION  Simple application sending order to set value in AO object,
//              based on IMS User API Object Access Services.
//              NOTE: AO object must be set in AUTO MODE in order to
//                    allow for change of VALUE.
//
// SYNTAX
*/
char setAOSyntax[] = “\
SYNTAX: setAO AOObjectName1 [AOObjectName2 ...] [newOutputValue]\n”;

/*
// -------------------- Standard C Includes
*/
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

/*
// -------------------- bciUserAPI Includes
*/
#include <bciUserAPI.h>
#include <bsUserAPI_def.h>

/*
// -------------------- AO object type Includes
*/
#include <omf_AO.h>
#include <dccodes.hh>

/*
// -------------------- DATA and CODE for application code example
*/
```

```
static const int Offset      = 128;

/*
// -------------------- Macros
*/
#define retStatToText(status) bciGetMessageText(BCI_CALL_STATUS,   Offset+status)
#define objStatToText(status) bciGetMessageText(BCI_OBJECT_STATUS, Offset+status)
#define opStatToText(status)  bciGetMessageText(BCI_OPERATION_STATUS, status-dcxbtxt_offset)

/*
// -------------------- Functions
*/
int isDigit(char* strPtr)
    {
    int                     digit = 1;
    while (*strPtr)
    {
    digit = (*strPtr == ‘-’ || *strPtr == ‘.’ || isdigit((int)*strPtr)) && digit;
    strPtr++;
    }
    return digit;
    }

/*
// -------------------- AO object data
*/
char**                  aoObjNames;
int                     aoOperation = OMF_AOORDERID;
size_t                  aoOperSize  = sizeof(Omf_AOORDER);
Omf_AOORDER**           aoOperOrder;

/*
// -------------------- MAIN CODE for application code example
*/
int main(int argc, char* argv[])
    {
    int                 objIdx;
    int                 request;
    int                 nbrOfObj      = argc-1;
    float               theNewAOValue = 12.34;
    BciStatus           result;
    BciObjStatus**      objStatus;

#ifdef __hpux
/*
// If HP environment: run constructor for C++ as program coded in C
*/
_main();
```

```
#endif

/* Check parameters */
if (argc > 1 && isDigit(argv[argc-1]))
{
nbrOfObj      = argc-2;
theNewAOValue = (float) atof(argv[argc-1]);
}
if (nbrOfObj == 0)
{
printf("%s", setAOSyntax);
exit(-1);
}

/* Allocate memory for AO object parameters */
aoObjNames  = (char**)          malloc(nbrOfObj*sizeof(char*));
objStatus   = (BciObjStatus**) malloc(nbrOfObj*sizeof(BciObjStatus*));
aoOperOrder = (Omf_AOORDER**)  malloc(nbrOfObj*sizeof(Omf_AOORDER*));
for (objIdx = 0; objIdx < nbrOfObj; objIdx++)
{
    objStatus[objIdx]   = malloc(sizeof(BciObjStatus));
    aoObjNames[objIdx]  = malloc(strlen(argv[1+objIdx]) + 1);
    strcpy(aoObjNames[objIdx], argv[1+objIdx]);
    aoOperOrder[objIdx] = (Omf_AOORDER*) malloc(sizeof(Omf_AOORDER));

/*
// set up the change value ORDER parameters
*/
aoOperOrder[objIdx]->opCode    = dcVALUE_CHANGE_DCX; /*
dcVALUE_CHANGE_DCX = 14 */
    aoOperOrder[objIdx]->opProp    = dcOUTPUT_DCX;      /* dcOUTPUT_DCX = 4        */
    aoOperOrder[objIdx]->typeOfReq = dcPROG;            /* dcPROG = 1              */
    aoOperOrder[objIdx]->eventLog  = 0;
    aoOperOrder[objIdx]->dialogId  = 0;
    aoOperOrder[objIdx]->aValue    = theNewAOValue;
    aoOperOrder[objIdx]->status    = 0;
    aoOperOrder[objIdx]->textIndex = 0;
}

printf("setAO: setting output value of %d AO objects to %5.2f ...\n",
nbrOfObj, theNewAOValue);

/* performing the ORDER command */
result = bciExecOperations(aoObjNames,
    nbrOfObj,
    aoOperation,
    aoOperSize,
    (void**)aoOperOrder,
    objStatus,
            0);
        if (result != bciSUCCESS)
    printf(“setAO: bciExecOperations()\n      ERROR %d (%s)\n”,
    result, retStatToText(result));
else
    for (objIdx = 0; objIdx < nbrOfObj; objIdx++)
    {
    if (objStatus[objIdx]->codes != bciOBJ_SUCCESS)
    printf(“setAO: bciGetAttributes(%s)\n      CODE %d (%s)\n”,
    aoObjNames[objIdx], objStatus[objIdx]->codes,
    objStatToText(objStatus[objIdx]->codes));
    else
    {
    if (aoOperOrder[objIdx]->status != 0)
    printf(“setAO: CHANGE ORDER to %s returned STATUS %d\n      TEXTINDEX %d
(%s)\n”,
     aoObjNames[objIdx], aoOperOrder[objIdx]->status,
    aoOperOrder[objIdx]->textIndex,
    opStatToText(aoOperOrder[objIdx]->textIndex));
        }
    }

/* Deallocate memory used */
for (objIdx = 0; objIdx < nbrOfObj; objIdx++)
{
free (aoOperOrder[objIdx]);
free (objStatus[objIdx]);
free (aoObjNames[objIdx]);
}
free (aoOperOrder);
free (objStatus);
free (aoObjNames);

exit(0);
}
```

The following is an example on how the command ORDER is executed from a User Object Operation towards an Analog Output object. The command changes the value of the AO object connected through the connection AOOut1. The library dccodes are assigned to the object type.

```
/* Declarations */
 IntSmall typeOfReq;
 gcBoolean eventLog;
 IntLong dialogId;
 Float aValue;
 IntLong dValue;
 IntSmall status;
 IntShort textIndex;
```

```
/* Using variables for clarity */
 typeOfReq = dcPROG;
 eventLog = gcFalse;
 dialogId = 0;
 status = 0;
 textIndex = 0;

/* Performing the ORDER command */

AOOut1.ORDER (dcVALUE_CHANGE_DCX,
              dcOUTPUT_DCX,
              typeOfReq,
              eventLog,
              dialogId,
              aValue,
              dValue,
              status,
              textIndex);
```

1.6.7 Event Section

1.6.7.1 General

The event declarations describe the events that other objects can be triggered by. By specifying a specific event with the Object Access Service **`bciGetAttribOnEvent`** the application program can get the requested data when the event occurs. The same applies when you configure User Objects, Reports and Calculations to be triggered by an event. When the event occurs the requesting application receives the requested data and is invoked for execution.

Which events that exist and when they occur is documented in the Event section.

Many events are defined without parameters. In that case they are documented as shown in Table 1-6 below. If any event has parameters they are documented in an additional table following the description table.

1.6.7.2 Events Available for the Majority of ABB Master Objects

The event **`EVENT`** is available for most ABB Master object types. It has no parameters. The exact nature of when it occurs varies from object type to object type. Thus, the reason for the event to trigger your application is described in the data sheet for each object type.:

Table 1-6. Example of EVENT description

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The normal case is that the VALUE bit of the STATUS attribute changes. (Toggles between 0 and 1). The event subscriptions of the DI objects always request for at least the STATUS attribute to be returned.

1.7 Messages to Events and Alarm Subscribers

The events described above are events used for triggering. The messages sent to the Event and Alarm subscribers (Event & Alarm), which are often referred to as events, are something completely different. Those messages are not a communication/triggering mechanism. They are simply messages to the Event & Alarm subscribers. They are sent from objects as a result of the following:

1. The object identifies a state which shall result in a message to the Event & Alarm subscribers. For example an alarm limit transgression. It verifies the content of the related event treatment pointer (called for example ERROR_TREAT, ALARM_TREAT). If the event treatment pointer is larger than 0 and the object event treatment is not blocked, a message is sent to the E&A subscribers.
2. For ABB Master Objects messages may be generated and sent to the Event & Alarm subscribers when the command ORDER is executed. This is specified with the command parameter *eventLog*, see the description in Table 1-5. When *eventLog* is true messages are generated. These messages inform that an Operator or a program has modified the object.

1.8 Error Messages

Error messages related to objects are either returned in the response from commands, see for example the parameters *status* and *textIndex* in Table 1-5, or results in error messages reported in IMS to the IMS system message log.

1.9 How to Use the Data Sheets

1.9.1 C/C++ Declarations Needed by Object Access Services

General

When you use the AdvaInform User API Object Access Services you need to know the C/C++ data types and constant names that describe the Object Types. For all object types a C include file has been generated from the OTDL definition. You find the C declarations through the path:

`/opt/advant/TypeDir/include/`

The declaration of each object type is available in an include file named **`omf_’ObjectTypeName’.h.`**

For the object type name AI, include the file **`/opt/advant/TypeDir/omf_AI.h`**

There are global data types that apply for all object types in the system and local data types specific for the object types. The global data types have the prefix Td. TdName is an example of a global data type. When your program is using the Object Access Services it may be necessary to refer to the C data types for global data types. They are available in the include file **`/opt/advant/TypeDir/include/omfAttributeT.h`**

Table 1-7. Example of commonly used global data types from omfAttributeT.h

OTDL data type	Description
TdName	Text string 20 character long + end mark. (char [21];)
TdDescription	Text string 28 character long + end mark. (char [29];)
TdStatus	32 bit long. When TdStatus is used there is a table in conjunction with the explained object type describing the usage of each bit. Note that when TdStatus is extended with a local data type declaration, the name of the C data type will be “objecttypeName”+TdStatus. It is available in the include file for the object type.

The C types corresponding to each attribute are defined with the following syntax:

`Omf_+’ObjectTypeName’+’Attributename’`

The Object Access Services refer to the object instances attributes, commands(operations) and events using their identities. The attribute, command, event and composite attributes id’s are declared as constants with the following syntax:

`OMF_+’ObjectTypeName’+’Attribute-/Command-/Event-/Composite Attribute-name’ID`

NOTE

The C declarations are all case sensitive, as are the C and C++ languages. If the object type name is BasicObject, the command Copy will get the id constant name:

`OMF_BasicObjectCopyID`

Examples:

The ID’s of the attribute VALUE, the Composite Attribute ain_ab1_dcx, the command ORDER and the event EVENT of the object type AI get the following constant names

`OMF_AIVALUEID, OMF_ain_ab1_dcxID, OMF_AIORDERID and OMF_AIEVENTID respectively.`

1.9.2 Access via SQL

This manual, *AdvaInform Object Types Reference Manual*, includes a description of attributes accessible using SQL access. Each object description in this manual includes an explanation of the attributes associated with the object type.

The IMS Relational Database as well as AdvaInform SQL*Connect - Client are covered in *AdvaInform Basic Function User's Guide*. This manuals describes access methods, privileges and configuration of AdvaInform SQL*Connect. Below is an introduction to access towards the Advant OCS using SQL.

1.9.2.1 IMS_USER

To allow access to the IMS Relational Database there is an IMS_USER (Oracle user) defined. No other Oracle user but IMS_USER is available for accessing the IMS Relational Database using SQL statements. You do not have access rights to DBA (Data Base Administrator).

The IMS_USER can be used when tools which support SQL*Net for TCP/IP is used. When starting **Station - Database access** or using AdvaInform Reports it is the IMS_USER that gets access to the IMS Relational Database.

The IMS_USER has read access to predefined database views. The views are towards Process Objects as well as System Application Objects. The attributes of the objects have corresponding columns in a view for each Object Type.

To list all tables/views available for IMS_USER:

1. Evoke **Station - Database Access** from the IMS menu.
2. Enter the SQL command:

```
SQL> SELECT table_name FROM IMS_TABLES;
```

To list all attributes/columns available for read from a table/view:

1. Choose **Station - Database Access** from the IMS menu.
2. Assuming that it is the table AI that is of interest. Enter the SQL command:

```
SQL> DESCRIBE AI;
```

NOTE

The Process Objects of type AI, AO, DI, DO, DAT and TEXT can be stored in the Master process station and the Advant Controller, or as Basic Objects in AdvaInform Object Handling. The AI, AO, DI, DO, DAT and TEXT views are related to AdvaInform SQL*Connect and are primarily to be used for Process Objects located in Advant Controllers. You may also add object instances of Basic Objects (included in AdvaInform Object Handling) to the AI, AO, DI, DO, DAT and TEXT views, but you do not have to, since the Basic Objects are accessible in USEROBJECTS.CO_AI_INST, CO_AO_INST.CO, CO_DI_INST, CO_DO_INST, CO_DAT_INST and CO_TEXT_INST, which have better performance.

1.9.2.2 Example on Access to a Process Object

AdvaInform SQL*Connect - Client makes the Advant OCS appear as a relational database. The Client is able to fetch data from a uniquely specified Object instance.

```
SELECT name, value, description FROM AI WHERE
NAME='AIC15_1';
```

The SELECT statement results in that the Core System tries to resolve the address to one Object instance AIC15_1. The result of the query will be name, value and description of the object instance.

```
NAME      VALUE     DESCRIPTION

AIC15_1   4.000     Calculated pressure
```

Figure 1-3. Example of result from SQL query.

1.9.3 Access from User Objects and Calculations

During the configuration you select what attribute to connect the input data connections to. In most cases this is self explanatory or well known. If not, read the attribute tables for the object type to determine which attribute to connect to.

When you program User Objects, you need to know the name of the attributes of the object connections. Furthermore, the syntax and parameters of the commands that you can use towards output connections must be described. This is all available in the Attribute and Command sections of each object type data sheet.

1.10 Restrictions

MasterPiece in Master network: Only object attributes that are included in the smallest composite attribute, or the smallest composite attribute itself, can be subscribed cyclically or on event from MasterPiece controllers. No restrictions for Advant Controllers or controllers in MOD 300 network.

For example: Only the STATUS, VALUE, NO_OF_DEC and ain_abl_dcx attributes from the AI object type can be subscribed cyclically or on event from MasterPiece controllers.

Chapter 2 Inheritance

2.1 General

A central function of Object Handling is the ability to inherit. Inheritance increases productivity and ease of maintenance and decreases errors. *AdvaBuild Object Type Builder User's Guide* gives a more detailed explanation on the features of inheritance. Since Object Types described in this manual use inheritance a short introduction on Basic inheritance is in its place.

2.2 Basic Inheritance

There are three pre-defined abstracts:

- A **BasicObject**, which has three defined attributes: name, description and status.
- A **DCSObject**, which has one defined attribute: name.
- A **BlankObject**, which does not include anything.

BasicObject automatically provides powerful functions and three essential attributes that all object types should have. It also provides the ability to send event messages to the Event and Alarm system. The BasicObject type is inherited by the IMS Basic Objects.

The **DCSObject** is primarily for ABB's internal use. Object types with limited functions from previous versions of ABB products are described using the DCSObject as a base.

Using **BlankObject** as inheritance means to inherit nothing. An object that inherits nothing cannot be programmed using Object Type Builder. The only reason to use BlankObject is to describe a *third-party* object for the core system.

The ready-made object types are **abstract**, which means that they can be inherited but not instantiated.

The BasicObject and the DCSObject do not include any connections. They support the following commands:

- Delete ()
- Deactivate ()
- Copy (Name)
- NormalOperation ()

An additional command which is supported by the object handlers and not by the object itself also needs to be explained:

- Activate (name)

Table 2-1. Global commands available for all IMS User Objects

Command	**Description**
Delete	Delete stops the execution of the object, see Deactivate below, and then deletes the object. All primary and disk memory resources that the object allocated are released. All references to the object are removed.
Deactivate	Deactivate stops the execution of the object. Its primary memory allocation is removed and it is no longer available in the run time environment.
Copy	Copy creates a copy of the object in question and gives it the given name. All other attributes are identical with the object from which it is copied.
NormalOperation	Execute the objects Normal Operation.
Activate (name)	To start execution of the object, execute the Object Handler command Activate. The named object’s configured information is read, the object is initialized in primary memory and the object’s execution is started.

Chapter 3 Process Objects

3.1 Analog Input - AI

3.1.1 How to Use

The Object Type Analog Input, here after called AI, exists as Basic Object type in AdvaInform Object Handling and as process object type in ABB Master Process Station (MasterPiece, Advant Controller).

Basic object: The AI object type gives limit checks and a strong event and alarm treatment with a large set of status information. The format type is floating point (single precision). It is therefore very suitable as input and output of calculations. It is also suitable to transport AI values from the Process Station and into IMS/Oracle.

The object type AI incorporates functions for:

- Input from optional Float Single connection.
- Storage of received input
- Limit checking
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

The AI objects have the following behavior depending on process Update Blocked and if the optional input connection is configured or not:

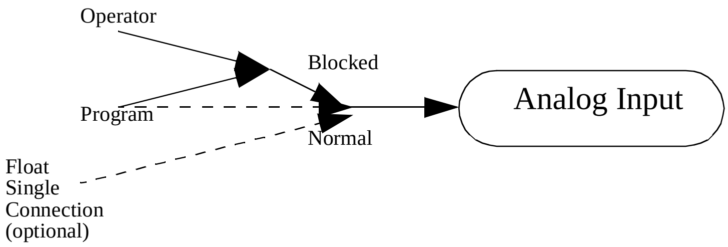


Figure 3-1. AI: Behavior in Normal and Blocked mode

1. The single input connection is configured.

   In normal mode the objects VALUE attribute can only be updated by the data read from the input connection. The operator may not modify the VALUE attribute. Programs may modify the VALUE but the update is ignored.

   In blocked mode it is allowed for an operator as well as for a program to modify the object attribute VALUE with commands. The input connection is disregarded.

2. The single input connection is not configured.

   In normal mode the objects VALUE attribute can only be updated by commands from programs.

   In blocked mode it is allowed for an operator as well as for a program to modify the object attribute VALUE with commands.

3.1.2 Overview

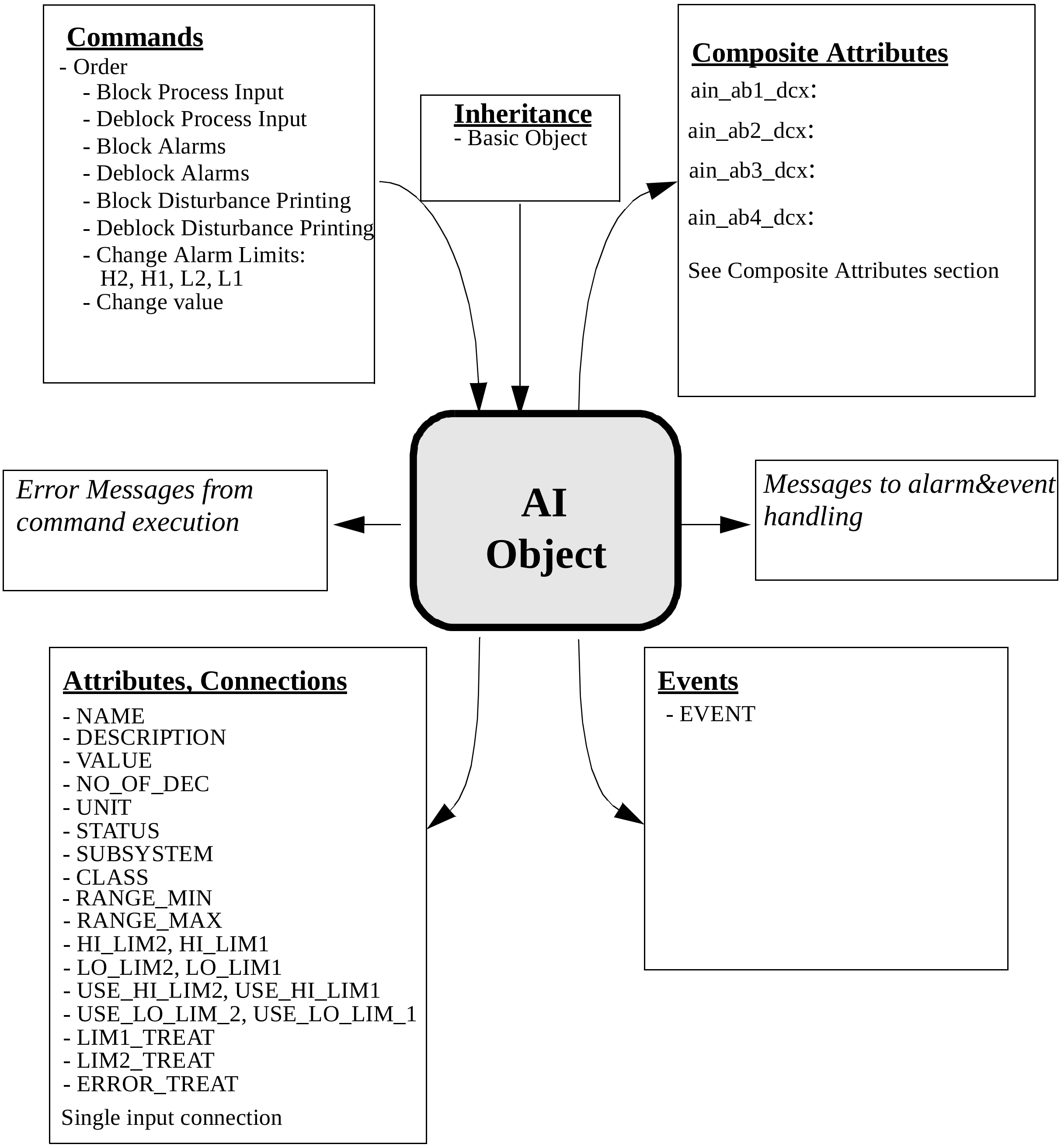


Figure 3-2. Overview of the Object Type Analog Input (AI)

3.1.3 Attributes

The object attributes are a subset of the corresponding attributes of the Analog Inputs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master data base element or Functional Unit manuals.

Table 3-1. Analog Input object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
VALUE	Float[12,4] 16,4	The current value of the object
NO_OF_DEC	IntSmall	Number of decimals
UNIT	String[7]	The unit of the value.
STATUS	TdStatus	Represents the status of the object. See Table 3-2
SUBSYSTEM	IntSmall	Process section 0 - 16
CLASS	IntSmall	The attribute has the value 0 in IMS
RANGE_MIN	Float[12,4] 16,4	The smallest allowed object value
RANGE_MAX	Float[12,4] 16,4	The highest allowed object value
HI_LIM 2	Float[12,4] 16,4	Alarm limit value, HI_LIM2, for object value.
HI_LIM1	Float[12,4] 16,4	Alarm limit value, HI_LIM1, for object value
LO_LIM1	Float[12,4] 16,4	Alarm limit value, LO_LIM1, for object value
LO_LIM2	Float[12,4] 16,4	Alarm limit value, LO_LIM2, for object value
USE_HI_LIM2	Boolean	Alarm limit HI_LIM2 used for object value
USE_HI_LIM1	Boolean	Alarm limit HI_LIM1 used for object value
USE_LO_LIM1	Boolean	Alarm limit LO_LIM1 used for object value
USE_LO_LIM2	Boolean	Alarm limit LO_LIM2 used for object value
LIM1_TREAT	IntShort	Value of event treatment definitions in MV/OS
LIM2_TREAT	IntShort	Value of event treatment definitions in MV/OS
ERROR_TREAT	IntShort	Value of event treatment definitions in MV/OS

Table 3-2. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run
error	2	Indicates the overview status of the object.
updated	3	Indicates if the object has been updated since object execution started.
UPD_BLK	4	Input from single connection blocked if true
MAN_ENTRY	5	Last update was from an operator/manual
SELECTED	6	Object is selected by an operator
NOR_OBJ_TREAT	7	(1)
H2_REP_FAIL_CTRL	8	(1)
ABOVE_HI_LIM2	9	Alarm limit transgressed
ABOVE_HI_LIM1	10	Alarm limit transgressed
BELOW_LO_LIM1	11	Alarm limit transgressed
BELOW_LO_LIM2	12	Alarm limit transgressed
ALARM_UNACK	13	Alarm unacknowledged
H1_REP_FAIL_CTRL	14	(1)
DISTURBANCE	15	Some kind of abnormal state exists
AIOVERFLOW	16	(1)
ALARM_BLK	17	Alarms are blocked
ALARM_PERIOD_BLK	18	(1)
PRINT_BLK	19	Alarm/Event printouts are blocked
L1_REP_FAIL_CTRL	20	(1)
LINKED	21	(1)
RELINK	22	(1)
NOERR_AT_OVF	23	(1)
TESTED	24	(1)
ACC_ERR	25	(1)
LOCKED	26	(1)
L2_REP_FAIL_CTRL	27	(1)
ER_REP_FAIL_CTRL	28	(1)

Table 3-2. TdStatus. Status bits (Continued)

Bits in TdStatus	Bit number	Description
REPEAT_FAIL_BLK	29	(1)
AI_ACTION	30	(1)
AI_RETYR	31	(1)
ERR_CTRL	32	(1)

(1) The status bit is not used in IMS Basic Objects

3.1.4 Connections

The AI has an optional Single Connection making it possible to read a float attribute from another object.

Table 3-3. Input connection

Name	Direction	Type/Format	Description
InputConnection	Input	Single/Float	Optional single input connection. Used as data input if configured.

3.1.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

ain_ab1_dcx:

STATUS, VALUE, NO_OF_DEC

ain_ab2_dcx:

STATUS, VALUE, NO_OF_DEC, RANGE_MAX, RANGE_MIN, NAME, DESCRIPTION, UNIT

ain_ab3_dcx:

STATUS, VALUE, NO_OF_DEC, RANGE_MAX, RANGE_MIN, HI_LIM2, HI_LIM1, LO_LIM1, LO_LIM2, NAME, DESCRIPTION, UNIT

ain_ab4_dcx:

STATUS, VALUE, NO_OF_DEC, RANGE_MAX, RANGE_MIN, HI_LIM2, HI_LIM1, LO_LIM1, LO_LIM2, NAME, DESCRIPTION, SUBSYSTEM, CLASS, LIM1_TREAT, LIM2_TREAT, UNIT)

3.1.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-4. AI operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command: Opcode OpProp**	**Functionality Output Exceptions**
Block input Block process input	dcBLOCK_DCX dcPROC_UPDATE_DCX	If not already blocked, the STATUS bit UPD_BLK is set to true. Blocks commands and Single Connection input.
Deblock input Deblock process input	dcDEBLOCK_DCX dcPROC_UPDATE_DCX	If blocked, the STATUS bit UPD_BLK is set to false
Alarm block Block disturbance alarm	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS bit ALARM_BLK is set to true and DISTURBANCE is set to false. Sends message to remove all the objects existing alarms from the alarm lists
Alarm deblock Deblock disturbance alarm	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked, the STATUS bit ALARM_BLK i set to false. If any alarm limit status is set,the STATUS bit DISTURBED is set to true. Check all alarm limit status bit, all that are true result in a message to the E&A subscribers.
Print block Block disturbance print	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS bit PRINT_BLK is set to true
Print deblock Deblock disturbance print	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, the STATUS bit PRINT_BLK is set to false.
H2 Lim Command for changing the Max Limit 2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the High limit 2 value. Must be within the range: RANGE_MAX and RANGE_MIN. Must also be larger than any other valid limit. Check value against alarm limits and generate message to the E&A subscribers if the new limit value causes any change in the state of the object.
L2 Lim Command for changing the Min Limit 2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the Low limit 2 value. Must be within the range: RANGE_MAX and RANGE_MIN. Must also be smaller than any other valid limit. Check value against alarm limits and generate message to the E&A subscribers if the new limit value causes any change in the state of the object.
H1 Lim Command for changing the Max Limit 1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the High limit 1 value. Must be within the range: RANGE_MAX and RANGE_MIN. Must also be smaller then H2 limit and larger then L1 limit Check value against alarm limits and generate message to the E&A subscribers if the new limit value causes any change in the state of the object.

Table 3-4. AI operation on the ORDER Command (Continued)

L1 Lim Command for changing the Min Limit 1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the Low limit 1 value. Must be within the range: RANGE_MAX and RANGE_MIN. Must also be larger then L2 limit and smaller then H1 limit. Check value against alarm limits and generate message to the E&A subscribers if the new limit value causes any change in the state of the object.
Command to change the value	dcVALUE_CHANGE_DCX dcACT_VALUE_DCX dcCALCVALUE_DCX	Req=Oper: Only allowed in blocked mode Req=Prog: Depends on mode and if single connection is configured. If the value is within the RANGE_MAX and RANGE_MIN then the new value is set. The value is checked against alarm limits and message to the E&A subscribers is generated if the new value causes any change in the state of the object.
Increase Increase output with 1%	dcINCR_PERCENT_DCX dcACT_VALUE_DCX dcCALCVALUE_DCX	Req=Oper: Only allowed in blocked mode Req=Prog: Depends on mode and if single connection is configured. Value is increased with 1% of total range. Checked against and limited to MAX_RANGE before updated. The value is checked against alarm limits and message to the E&A subscribers is generated if the new value causes any change in the state of the object.
Decrease Decrease output with 1%	dcDECR_PERCENT_DCX dcACT_VALUE_DCX dcCALCVALUE_DCX	Req=Oper: Only allowed in blocked mode Req=Prog: Depends on mode and if single connection is configured. Value is decreased with 1% of total range. Checked against and limited to MIN_RANGE before updated. The value is checked against alarm limits and message to the E&A subscribers is generated if the new value causes any change in the state of the object.

3.1.7 Events

The AI object type has one event called EVENT:

Table 3-5. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the AI objects always request for at least the STATUS attribute to be returned.

3.1.8 SQL Views

Table 3-6. Description AI

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
STATUS	NOT NULL	NUMBER(10)
UNIT		VARCHAR2(6)
RANGE_MIN	NOT NULL	FLOAT(24)
RANGE_MAX	NOT NULL	FLOAT(24)
VALUE	NOT NULL	FLOAT(24)
NO_OF_DEC	NOT NULL	NUMBER(3)
SUBSYSTEM	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
HI_LIM2	NOT NULL	FLOAT(24)
HI_LIM1	NOT NULL	FLOAT(24)
LO_LIM1	NOT NULL	FLOAT(24)
LO_LIM2	NOT NULL	FLOAT(24)
LIM1_TREAT	NOT NULL	NUMBER(5)
LIM2_TREAT	NOT NULL	NUMBER(5)
USE_HI_LIM2		NUMBER
USE_HI_LIM1		NUMBER
USE_LO_LIM1		NUMBER
USE_LO_LIM2		NUMBER
ERROR_TREAT		NUMBER

Table 3-7. Bit-presentation STATUS, AI

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
ACTIVE		NUMBER
ERROR		NUMBER
UPDATED		NUMBER
UPD_BLK		NUMBER
MAN_ENTRY		NUMBER
SELECTED		NUMBER
NOR_OBJ_TREAT		NUMBER
H2_REP_FAIL_CTRL		NUMBER
ABOVE_HI_LIM2		NUMBER
ABOVE_HI_LIM1		NUMBER
BELOW_LO_LIM1		NUMBER
BELOW_LO_LIM2		NUMBER
ALARM_UNACK		NUMBER
H1_REP_FAIL_CTRL		NUMBER
DISTURBANCE		NUMBER
OVERFLOW		NUMBER
ALARM_BLK		NUMBER
ALARM_PERIOD_BLK		NUMBER
PRINT_BLK		NUMBER
L1_REP_FAIL_CTRL		NUMBER
LINKED		NUMBER
RELINK		NUMBER
NOERR_AT_OVF		NUMBER
TESTED		NUMBER
ACC_ERR		NUMBER
LOCK1		NUMBER
L2_REP_FAIL_CTRL		NUMBER

Table 3-7. Bit-presentation STATUS, AI (Continued)

Name	Null?	Type
ER_REP_FAIL_CTRL		NUMBER
REPEAT_FAIL_BLK		NUMBER
AI_ACTION		NUMBER
AI_RETRY		NUMBER
ERR_CTRL		NUMBER

3.2 Analog Output - AO

3.2.1 How to Use

The Object Type Analog Output, here after called AO, exists as Basic Object type in AdvaInform Object Handling and as process object in ABB Master Process Station.

AO's are suitable as output for certain types of calculations and as output objects for operator interaction with a set of commands adapted to output control. It is also suitable for the transfer of floating point data from IMS (user objects/DB tables) to any AO object residing in another node (normally Process Station).

The object type Analog Output incorporates functions for:

- Output attribute storage and updating
- Output to optional AO object connection (normally in Process Station)
- Input from optional Single Connection (Floating point)
- Limit checking prior to storage
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Services to treat Commands
- Auto and Manual modes of control
- Update of values at system initialization

The AO object consists of a database instance for each object, for storage of inputs and various other descriptive properties and control parameters.

It has an optional input single connection of data type float. If not configured it has no effect on the object types behavior.

It also has an optional AO output object connection. If not configured it has no effect on the object types behavior.

The AO object type controls the output (value/command) and forces the value to be within given limits. It has a limited event and alarm handling with a limited set of status information. The format type is floating point (single precision).

The AO object behaves in different ways depending on if the optional input and output connection respectively are configured or not.

- If the Single Connection input is configured and the object is in AUTO mode, the data read from that input always overrides the values that are set with commands to the objects.
- If the AO output object connection is configured a command will be sent to the connected AO object as soon as the value attribute is modified.

The AO objects have the following behavior in MAN and AUTO mode:

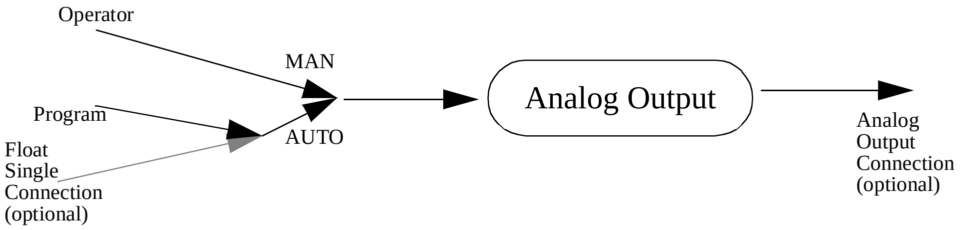


Figure 3-3. AO: Behavior in MAN/AUTO mode

- In AUTO mode the objects VALUE attribute can only be updated by commands from a program or by the data read from the input connection. The operator may not modify the VALUE attribute.
- In MAN mode it is allowed for an operator to modify the object attribute VALUE but not for a program. The input connection is disregarded.

3.2.2 Overview

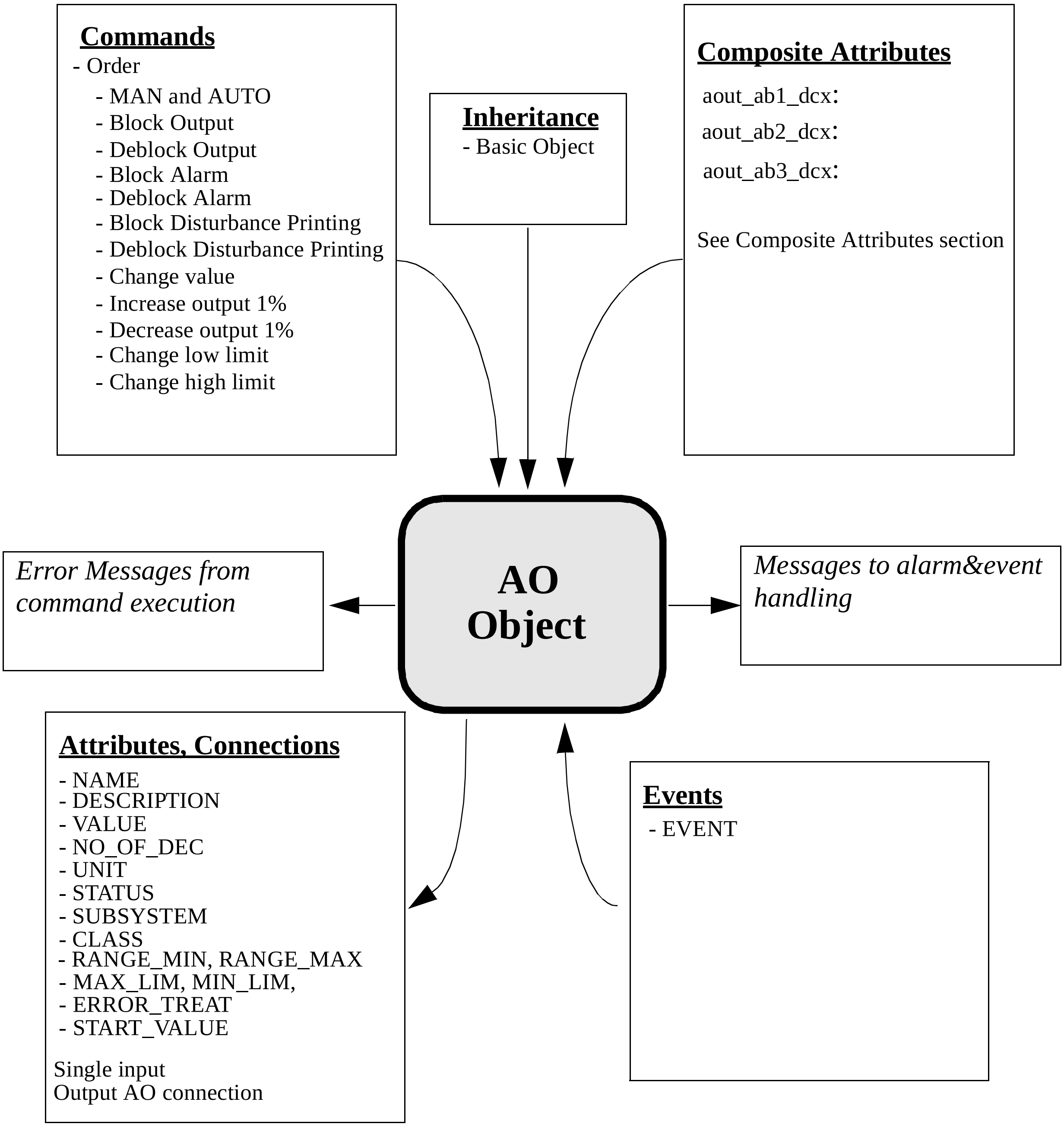


Figure 3-4. Overview of the Object Type AO

3.2.3 Attributes

The object attributes are a subset of the corresponding attributes of the Analog Output residing in the ABB Master Process Station.

Table 3-8. AO object type attributes

Attribute	OTDL Data Type	Description
NAME	Tdname	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
VALUE	Float[12,4] 16,4	The current value of the object
NO_OF_DEC	IntSmall	Number of decimals
UNIT	String[7]	The unit of the value.
STATUS	TdStatus	Represents the status of the object. See Table 3-9
SUBSYSTEM	IntSmall	Process section 0 - 16
CLASS	IntSmall	The attribute has the value 0 in IMS
RANGE_MIN	Float[12,4] 16,4	The lowest allowed value
RANGE_MAX	Float[12,4] 16,4	The highest allowed value
MAX_LIM	Float[12,4] 16,4	The highest allowed value given as a command
MIN_LIM	Float[12,4] 16,4	The lowest allowed value given as a command
ERROR_TREAT	IntShort	Value of event treatment definitions in MV/OS
START_VALUE	IntShort	Not used in IMS

Table 3-9. TdStatus. AO Status bits

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run
error	2	Indicates the overview status of the object.
updated	3	Indicates if the object has been updated since object execution started.
OUTP_BLK	4	Output to output connection blocked
SELECTED	6	Object is selected by an operator
NOR_OBJ_TREAT	7	(1)
MAN_MODE	8	Object in manual mode, only operator input accepted

Table 3-9. TdStatus. AO Status bits (Continued)

Bits in TdStatus	Bit number	Description
ON_MAX_LIM	9	(1)
ON_MIN_LIM	12	(1)
ALARM_UNACK	13	Alarm unacknowledged
ALARM_BLK	17	Alarms are blocked
AL_PERIOD_BLK	18	(1)
PRINT_BLK	19	Alarm/Event printouts are blocked
OUTP_RESTART	20	(1)
LOCKED	21	(1)
USE_MAX_LIM	22	(1)
USE_MIN_LIM	23	(1)
TESTED	24	(1)

(1) Not used in IMS Basic Object

3.2.4 Connections

The AO has an optional Single Connection making it possible to read a float attribute from another object:

Table 3-10. Input connection

Name	Direction	Type/Format	Description
InputConnection	Input	Single/Float	Optional single input connection. Used as data input if configured.

The AO has an optional output AO object connection making it possible to update a boolean attribute from another object:

Table 3-11. Output connection

Name	Direction	Type/Format	Description
OutputConnection	Output	Object/AO	Optional output AO connection. If configured, the output AO is updated with the new VALUE each time the value changes.

3.2.5 Composite Attributes

The following Composite Attributes exist. The attribute list behind the Composite Attribute name describes what attributes that are included.

aout_ab1_dcx:

STATUS, VALUE, NO_OF_DEC

aout_ab2_dcx:

STATUS, VALUE, NO_OF_DEC, RANGE_MAX, RANGE_MIN, NAME, DESCRIPTION, UNIT

aout_ab3_dcx: (

STATUS, VALUE, NO_OF_DEC, RANGE_MAX, RANGE_MIN, NAME, DESCRIPTION, SUBSYSTEM, CLASS, MAX_LIM, MIN_LIM, UNIT

3.2.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command:

Table 3-12. AO Object Type operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command: Opcode OpProp**	**Functionality Output Exceptions**
MAN Set to Manual mode	dcMANUAL_DCX -	If not already manual, set the STATUS bit MAN to true. Blocks program and single connection input updates.
AUTO Set to Automatic mode	dcAUTO_DCX -	If not already auto, set the STATUS bit MAN to false.
Block output Block process output	dcBLOCK_DCX dcOUTPUT_DCX	If not already blocked, set STATUS bit OUTP_BLK to true.
Deblock output Deblock process output	dcDEBLOCK_DCX dcOUTPUT_DCX	If blocked, set the STATUS bit OUTP_BLK to false.
Block Alarm Block disturbance alarm	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, set the STATUS bit ALARM_BLK to true.
Deblock Alarm Deblock disturbance alarm	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked, set the STATUS bit ALARM_BLK to false.
Block Print Block disturbance printing	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, set the STATUS bit PRINT_BLK to true.

Table 3-12. AO Object Type operation on the ORDER Command (Continued)

Deblock Print Deblock disturbance printing	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, set the STATUS bit PRINT_BLK to false.
H Lim Command for changing the Max Limit	dcVALUE_CHANGE_DCX dcMAXLIM_DCX	Changes the MAX_LIM value. Must be larger then MIN_LIM and within RANGE_MAX - RANGE_MIN.
L Lim Command for changing the Min Limit	dcVALUE_CHANGE_DCX dcMINLIM_DCX	Changes the MIN_LIM value. The new value must be smaller then MAX_LIM and within RANGE_MAX and RANGE_MIN.
Command to change the value	dcVALUE_CHANGE_DCX dcOUTPUT_DCX dcCALCVALUE_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured. Changes the VALUE. The new value must be within RANGE_MAX - RANGE_MIN and within MAX_LIM - MIN_LIM.
Increase Increase output with 1%	dcINCR_PERCENT_DCX -	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured, Value is increased with 1% of total range. The new value is limited by MAX_LIM.
Decrease Decrease output with 1%	dcDECR_PERCENT_DCX -	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured, Value is decreased with 1% of total range. The new value is limited by MIN_LIM.

3.2.7 Events

The AO object type has one event called EVENT:

Table 3-13. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the AO objects always request for at least the STATUS attribute to be returned.

3.2.8 SQL Views

Table 3-14. Description AO

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
STATUS	NOT NULL	NUMBER(10)
UNIT		VARCHAR2(6)
VALUE	NOT NULL	FLOAT(24)
NO_OF_DEC	NOT NULL	NUMBER(3)
RANGE_MIN	NOT NULL	FLOAT(24)
RANGE_MAX	NOT NULL	FLOAT(24)
MAX_LIM	NOT NULL	FLOAT(24)
MIN_LIM	NOT NULL	FLOAT(24)
SUBSYSTEM	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
START_VALUE		NUMBER
ERROR_TREAT		NUMBER

Table 3-15. Bit presentation STATUS, AO

Name	Null?	Type
NAME		VARCHAR2(20)
ACTIVE		NUMBER
ERROR		NUMBER
UPDATED		NUMBER
OUTP_BLK		NUMBER
BIT5		NUMBER
SELECTED		NUMBER
NOR_OBJ_TREAT		NUMBER
MAN_MODE		NUMBER

Table 3-15. Bit presentation STATUS, AO (Continued)

Name	Null?	Type
ON_MAX_LIM		NUMBER
BIT10		NUMBER
BIT11		NUMBER
ON_MIN_LIM		NUMBER
ALARM_UNACK		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
ALARM_BLK		NUMBER
AL_PERIOD_BLK		NUMBER
PRINT_BLK		NUMBER
OUTP_RESTART		NUMBER
LOCK1		NUMBER
USE_MAX_LIM		NUMBER
USE_MIN_LIM		NUMBER
TESTED		NUMBER
BIT25		NUMBER
BIT26		NUMBER
LINKED		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

3.3 Digital Input - DI

3.3.1 How to Use

The Object Type Digital Input, here after called DI, exists as Basic Object in AdvaInform Object Handling and as object in ABB Master Process Station.

The DI object type has a strong event and alarm handling and is suitable for storage of digital in and out information to calculations and various applications. It is also suitable to transport DI values from the Process Station and into IMS Relational Database.

The object type DI incorporates functions for:

- Input from optional boolean Single connection.
- Storage of received input
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands

The DI objects have the following behavior depending on process Update Blocked and if the optional input connection is configured or not.

1. The single input connection is configured.

   In normal mode the objects VALUE attribute can only be updated by the data read from the input connection. The operator may not modify the VALUE attribute. Programs may modify the VALUE but the update is ignored.

   In blocked mode it is allowed for an operator as well as for a program to modify the object attribute VALUE with commands. The input connection is disregarded.

2. The single input connection is not configured.

   In normal mode the objects VALUE attribute can only be updated by commands from programs.

   In blocked mode it is allowed for an operator as well as for a program to modify the object attribute VALUE with commands.

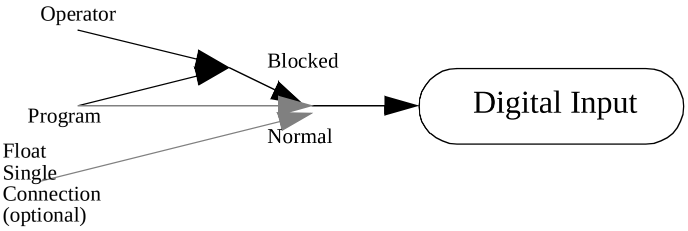


Figure 3-5. DI: Behavior in Normal and Blocked mode

3.3.2 Overview

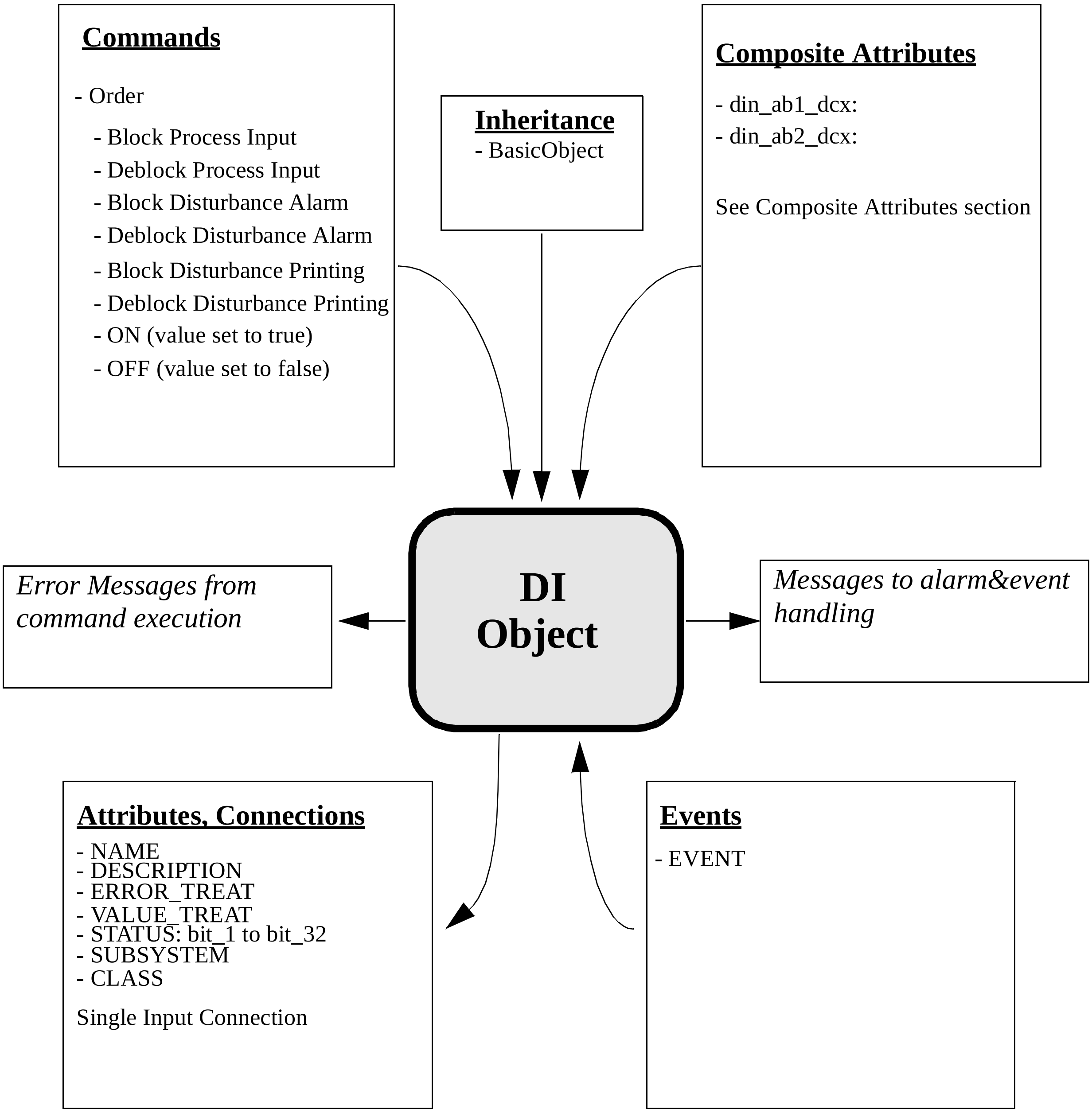

Commands
- Order
- Block Process Input
- Deblock Process Input
- Block Disturbance Alarm
- Deblock Disturbance Alarm
- Block Disturbance Printing
- Deblock Disturbance Printing
- ON (value set to true)
- OFF (value set to false)
Inheritance
- BasicObject
Composite Attributes
- din_ab1_dcx:
- din_ab2_dcx:
See Composite Attributes section
Error Messages from command execution
DI Object
Messages to alarm&event handling
Attributes, Connections
- NAME
- DESCRIPTION
- ERROR_TREAT
- VALUE_TREAT
- STATUS: bit_1 to bit_32
- SUBSYSTEM
- CLASS
Single Input Connection
Events
- EVENT

Figure 3-6. Overview of the Object Type DI.

3.3.3 Attributes

The object attributes described are a subset of the corresponding attributes of the Digital Inputs residing in the ABB Master Process Station.

Table 3-16. Digital Input (DI) object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName(c)	The name of the object.
DESCRIPTION	TdDescription (c)	The description of the object.
ERROR_TREAT	IntShort(c)	The event treatment number in the Operator Station / MasterView. Determines how the error status shall be processed and presented in the event/alarm lists, as a result of a value change. If set to 0, no event will be generated.
VALUE_TREAT	IntShort(c)	The event treatment number in the Operator Station / MasterView. Determines how events and alarms shall be processed and presented in the event/alarm lists, as a result of a value change. If set to 0, no event will be generated.
STATUS	TdStatus	Represents the status of the object. See Table 3-17
SUBSYSTEM	IntSmall(c)	Process section 0 - 16. If 0, the objects events are not sorted at all. If set to 1 to 16, events from the object are sorted into the corresponding process section.
CLASS	IntSmall(c)	The attribute has the value 0 in IMS

Table 3-17. TdStatus. DI status bits.

Bits in TdStatus	Bit number	Description
active	1	If TRUE, the object is active. If false, the object is stopped.
error	2	If TRUE, the object is in an error state. This indicates a serious malfunction.
updated	3	If TRUE, the object is updated (has executed) since start-up of the node it is executing in.
UPD_BLK	4	Input from single connection blocked if true The DI objects have the following behavior depending on process Update Blocked: - In normal mode it is allowed to update the object attribute VALUE from a Program if the input connection is not configured. - In blocked mode it is allowed for an operator or program to modify the object attribute VALUE.

Table 3-17. TdStatus. DI status bits. (Continued)

Bits in TdStatus	Bit number	Description
MAN_ENTRY	5	If TRUE, the last update of the object came from an operator, otherwise from a program.
SELECTED	6	The object is selected from an operator display.
NOR_OBJ_TREAT	7	(1)
ERR_CTRL	8	(1)
VALUE	9	The current value 1 or 0.
SEC_VALUE	10	(1)
NORM_POSN	11	(1)
SEC_NORM_POSN	12	(1)
ALARM_UNACK	13	The object has an unacknowledged alarm
REP_FAIL_BLK	14	(1)
DISTURBANCE	15	The object is in an error state
CALC_VALUE	16	(1)
ALARM_BLK	17	Alarm reporting is blocked
AL_PERIOD_BLK	18	(1)
PRINT_BLK	19	Event and alarm printouts are blocked.
REP_FAIL_CTRL	20	(1)
LOCKED	21	(1)
INVERTED	22	(1)
DUAL_IND	23	(1)
TESTED	24	(1)
V9_CONTROL	26	(1)
LINKED	27	(1)
DI_RETRY	28	(1)
DIC_IND_RED	29	(1)
DI_ACTION	30	(1)
DIC_IND_YELLOW	31	(1)
DIC_IND_GREEN	32	(1)

(1) The attribute is (1) for the Basic Object Type in AdvaInform Object Handling.

3.3.4 Connections

The DI has an optional Single Connection making it possible to read a boolean attribute from another object:

Table 3-18. Input connection

Name	Direction	Type/Format	Description
InputConnection	Input	Single / Boolean	Optional boolean single input connection Used as input to the STATUS.VALUE attribute if configured

3.3.5 Composite Attributes

The following Composite Attributes exist. The attribute list behind the Composite Attribute name describes what attributes that are included.

din_ab1_dcx:

STATUS

din_ab2_dcx:

STATUS, NAME, DESCRIPTION, SUBSYSTEM, CLASS,VALUE_TREAT

3.3.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-19. The DI operation on the ORDER Command

Type of function for corresponding parameters.	**Parameters to the ORDER command: Opcode OpProp**	**Functionality Output Exceptions**
Block process input	dcBLOCK_DCX dcPROC_UPDATE_DCX	If not already blocked, then the STATUS bit UPD_BLK is set to true
Deblock process input	dcDEBLOCK_DCX dcPROC_UPDATE_DCX	If blocked, then the STATUS bit UPD_BLK is set to false
Block disturbance alarm	dcBLOCK_DCX dcDIST_ALARM_DCX dcDUMMY_RROP_DCX	If not already blocked, then the STATUS bit ALARM_BLK is set to true
Deblock disturbance alarm	dcDEBLOCK_DCX dcDIST_ALARM_DCX dcDUMMY_RROP_DCX	If blocked, then the STATUS bit ALARM_BLK is set to false
Block disturbance printing	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, then the STATUS bit PRINT_BLK is set to true
Deblock disturbance printing	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, set STATUS bit PRINT_BLK to false
ON	dcVALUE_CHANGE_DCX dcACT_VALUE_DCX dcCALCVALUE_DCX (dValue = 1)	Req=Oper: Only allowed in blocked mode Req=Prog: Only allowed in blocked mode if configured single connection
OFF	dcVALUE_CHANGE_DCX dcACT_VALUE_DCX dcCALCVALUE_DCX (dValue = 0)	Req=Oper: Only allowed in blocked mode Req=Prog: Only allowed in blocked mode if configured single connection

3.3.7 Events

The DI object type has one event called EVENT:

Table 3-20. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The normal case is that the VALUE bit of the STATUS attribute changes. (Toggles between 0 and 1). The event subscriptions of the DI objects always request for at least the STATUS attribute to be returned.

3.3.8 SQL Views

Table 3-21. Description DI

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
STATUS	NOT NULL	NUMBER(10)
SUBSYSTEM	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
VALUE_TREAT		NUMBER
ERROR_TREAT		NUMBER

Table 3-22. Bit presentation STATUS, DI

Name	Null?	Type
NAME		VARCHAR2(20)
ACTIVE		NUMBER
ERROR		NUMBER
UPDATED		NUMBER
UPD_BLK		NUMBER
MAN_ENTRY		NUMBER
SELECTED		NUMBER
NOR_OBJ_TREAT		NUMBER

Table 3-22. Bit presentation STATUS, DI (Continued)

Name	Null?	Type
ERR_CTRL		NUMBER
VALUE		NUMBER
SEC_VALUE		NUMBER
NORM_POSN		NUMBER
SEC_NORM_POSN		NUMBER
ALARM_UNACK		NUMBER
REP_FAIL_BLK		NUMBER
DISTURBANCE		NUMBER
CALC_VALUE		NUMBER
ALARM_BLK		NUMBER
ALARM_PERIOD_BLK		NUMBER
PRINT_BLK		NUMBER
REP_FAIL_CTRL		NUMBER
LOCK1		NUMBER
INVERTED		NUMBER
DUAL_IND		NUMBER
TESTED		NUMBER
BIT25		NUMBER
V9_CONTROL		NUMBER
LINKED		NUMBER
DI_RETRY		NUMBER
DIC_IND_RED		NUMBER
DI_ACTION		NUMBER
DIC_IND_YELLOW		NUMBER
DIC_IND_GREEN		NUMBER

3.4 Digital Output - DO

3.4.1 How to Use

The Object Type Digital Output, here after called DO, exists as Basic Object type in AdvaInform Object Handling and as process object type in ABB Master Process Station.

DO’s are suitable as digital output for certain types of calculations and as output objects for operator interaction with a set of commands adapted to output control. It is also suitable for the transfer of boolean data from IMS to any DO object residing in another node (normally process Station).

The object type Digital Output incorporates functions for:

- Output (attribute) value storage and updating
- Output to optional DO object connection (normally in Process Station)
- Input from connected (optional) single connection of type boolean.
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Services to treat Commands
- Auto and Manual modes of control
- Update of values at system initialization

It has an optional input single connection of data type boolean. If not configured it has no effect on the object types behavior.

It also has an optional DO output object connection. If not configured it has no effect on the object types behavior.

The DO object behaves in different ways depending on if the optional input and output connection respectively are configured or not.

- If the Single Connection input is configured and the object is in AUTO mode, the data read from that input always overrides the values that are set with commands to the objects.
- If the DO output object connection is configured a command will be sent to the connected DO object as soon as the value attribute is modified.

The DO objects have the following behavior in MAN and AUTO mode:

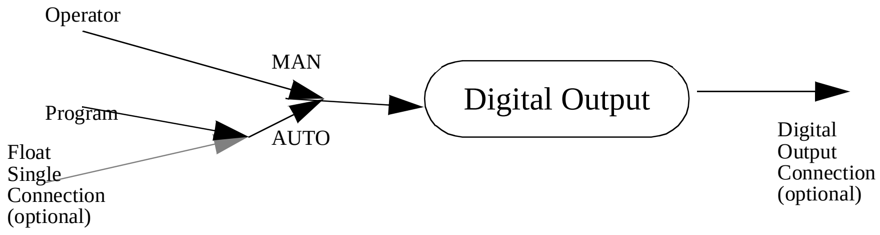


Figure 3-7. DO: Behavior in MAN/AUTO mode

- In AUTO mode the objects STATUS.VALUE attribute can only be updated by commands from a program or by the data read from the input connection. The operator may not modify the STATUS.VALUE attribute.
- In MAN mode it is allowed for an operator to modify the object attribute STATUS.VALUE but not for a program. The input connection is disregarded.

The object type has a limited event and alarm handling with a limited set of status information.

3.4.2 Overview

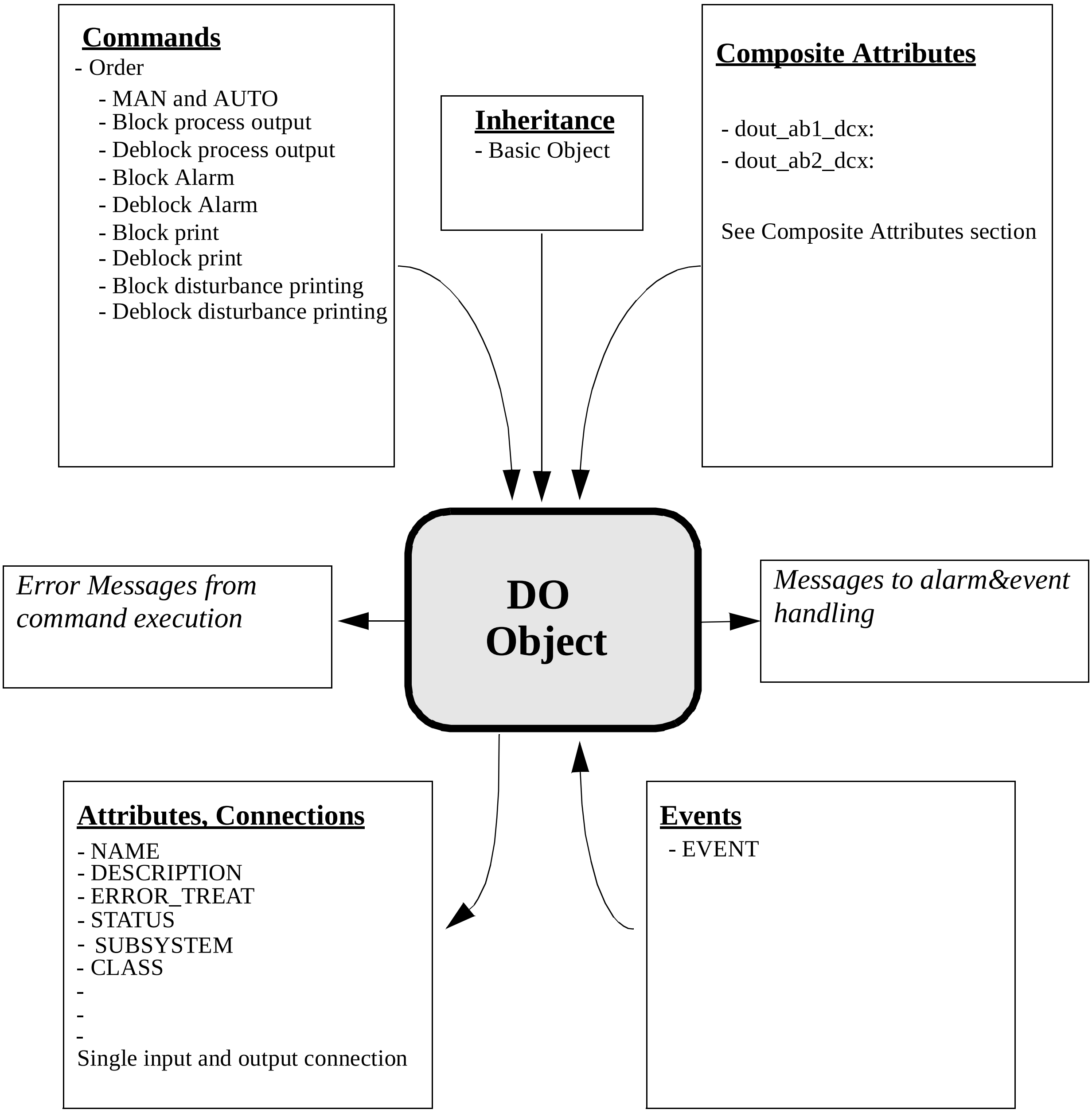


Figure 3-8. Overview of the Object Type DO

3.4.3 Attributes

The object attributes are a subset of the corresponding attributes of the Digital Output residing in the ABB Master Process Station.

Table 3-23. Digital Output object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescrition	The description of the object.
ERROR_TREAT	IntShort	Value of event treatment definitions in MV/OS
STATUS	TdStatus	Represents the status of the object. See Table 3-24
SUBSYSTEM	IntSmall	Process section 0 - 16
CLASS	IntSmall	The attribute has the value 0 in IMS

Table 3-24. TdStatus. DO Status bits

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run
error	2	Indicates the overview status of the object.
updated	3	Indicates if the object has been updated since object execution started.
OUTP_BLK	4	Output to output connection blocked
SELECTED	6	Object is selected by an operator
NOR_OBJ_TREAT	7	(1)
MAN_MODE	8	Object in manual mode, only operator input accepted
VALUE	9	Current value
START_VALUE	11	(1)
ALARM_UNACK	13	Alarm unacknowledged
ALARM_BLK	17	Alarms are blocked
AL_PERIOD_BLK	18	(1)
PRINT_BLK	19	Alarm/Event printouts are blocked
OUTP_RESTART	20	(1)
LOCKED	21	(1)

Table 3-24. TdStatus. DO Status bits (Continued)

Bits in TdStatus	Bit number	Description
INVERTED	22	(1)
TESTED	24	(1)
ORDER_TO	25	(1)
ORDER_FROM	26	(1)
LINKED	27	(1)

3.4.4 Connections

The DO has an optional Single Connection making it possible to read a boolean attribute from another object:

Table 3-25. DO Input connection

Name	Direction	Type/Format	Description
InputConnection	Input	Single / Boolean	Optional boolean single input connection Used as input to the STATUS.VALUE attribute if configured

The DO has an optional output DO object connection making it possible to update a boolean attribute from another object:

Table 3-26. DO Output connection

Name	Direction	Type/Format	Description
OutputConnection	Output	Object/ DO	Optional output DO connection. If configured, the output DO is updated with the new STATUS.VALUE each time the value changes.

3.4.5 Composite Attributes

The following Composite Attributes exist. The attribute list behind the Composite Attribute name describes what attributes that are included.

dout_ab1_dcx:

STATUS

dout_ab2_dcx:

STATUS, NAME, DESCRIPTION, SUBSYSTEM, CLASS

3.4.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-27. Object Type DO operation on the Order Command

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
MAN Set to Manual mode	dcMANUAL_DCX -	If not already manual, the STATUS bit MAN_MODE is set to true.
AUTO Set to Automatic mode	dcAUTO_DCX -	If not already auto, the STATUS bit MAN_MODE is set to false.
Block output Block process output	dcBLOCK_DCX dcOUTPUT_DCX	If not already blocked, the STATUS bit OUTP_BLK is set to true
Deblock output Deblock process output	dcDEBLOCK_DCX dcOUTPUT_DCX	If blocked, the STATUS bit OUTP_BLK is set to false
Block Alarm Block disturbance alarm	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS bit ALARM_BLK is set to true.
Deblock Alarm Deblock disturbance alarm	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked, the STATUS bit ALARM_BLK is set to false.
Block print Block disturbance printing	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS bit PRINT_BLK is set to true.
Deblock print Deblock disturbance printing	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, the STATUS bit PRINT_BLK is set to false.

Table 3-27. Object Type DO operation on the Order Command (Continued)

ON	dcONOPENSTART_DCX -	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Will be overridden if single connection input exists. The value is set to true
OFF	dcOFFCLOSE_DCX -	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Will be overridden if single connection input exists. The value is set to true

3.4.7 Events

The DO object type has one event called EVENT:

Table 3-28. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The normal case is that the VALUE bit of the STATUS attribute changes. (Toggles between 0 and 1). The event subscriptions of the DO objects always request for at least the STATUS attribute to be returned.

3.4.8 SQL Views

Table 3-29. Description DO

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
STATUS	NOT NULL	NUMBER(10)
SUBSYSTEM	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
ERROR_TREAT		NUMBER

Table 3-30. Bit presentation STATUS, DO

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
ACTIVE		NUMBER
ERROR		NUMBER
UPDATED		NUMBER
OUTP_BLK		NUMBER
BIT5		NUMBER
SELECTED		NUMBER
NOR_OBJ_TREAT		NUMBER
MAN_MODE		NUMBER
VALUE		NUMBER
BIT10		NUMBER
START_VALUE		NUMBER
BIT12		NUMBER
ALARM_UNACK		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
ALARM_BLK		NUMBER
AL_PERIOD_BLK		NUMBER
PRINT_BLK		NUMBER
OUTP_RESTART		NUMBER
LOCK1		NUMBER
INVERTED		NUMBER
BIT23		NUMBER
TESTED		NUMBER
ORDER_TO		NUMBER
ORDER_FROM		NUMBER
LINKED		NUMBER

Table 3-30. Bit presentation STATUS, DO (Continued)

Name	Null?	Type
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

3.5 DAT Object - DAT

3.5.1 How to Use

The Object Type DAT exists as Basic Object type in AdvaInform Object Handling and as process object type in ABB Master Process Station.

The DAT object type gives four different formats. The floating point, integer and integer word format are suitable for in and output to calculations and applications without any need for status information and with an uncomplicated behavior in all situations (no MAN or AUTO mode, always accessible). The DAT objects have a limited set of commands available. There is no event and alarm treatment.

The object type DAT incorporates functions for:

- Storage of received input (calculated)
- Support for subscription requests (on demand, 3 cyclic intervals)
- Services to treat Commands

The format type Boolean 32 is very useful for storage of sets of boolean/bit information. Only the first boolean can be altered through commands.

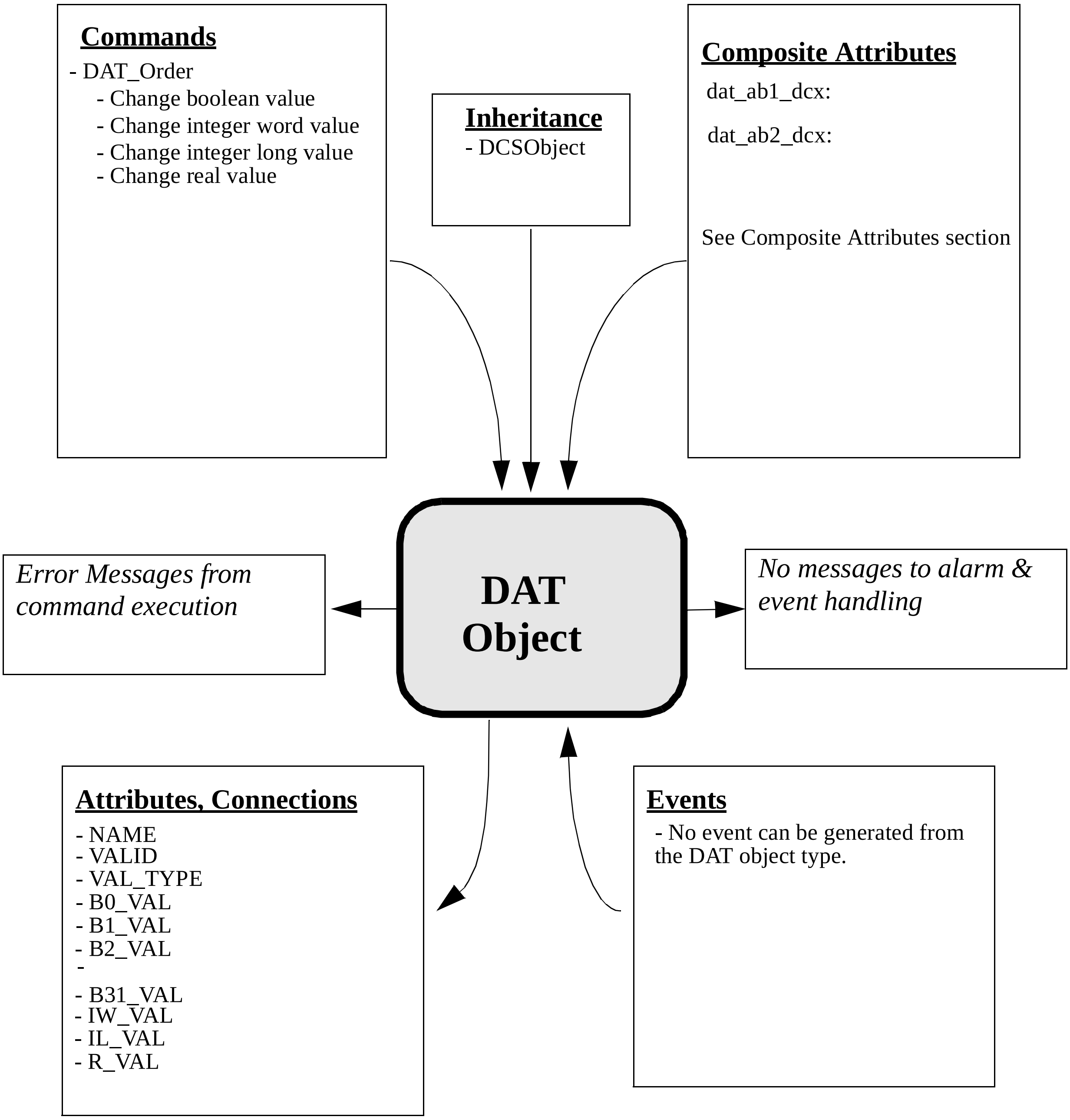
Basic Objects store the value in all 4 formats no matter what the real format is. This is because Oracle does not support union data types. If a user makes a SQL SELECT statement all data parts are correct even though only one corresponds to the data type of the object instance. The data is made identical on “bit basis”. This means that a floating point value does not make sense in integer format and vice versa.

The **DAT** object type formats are determined by the **VAL_TYPE** attribute:

Table 3-31. VAL_TYPE Value and Corresponding Format

VAL_TYPE value	Format	Valid value attribute
1	Boolean 32	B0_VAL B1_VAL .... B31_VAL
2	Integer word	I_VAL
3	Integer long	IL_VAL
4	Floating point	R_VAL

3.5.2 Overview



Commands

- DAT_Order
  - Change boolean value
  - Change integer word value
  - Change integer long value
  - Change real value

Inheritance

- DCSObject

Composite Attributes

dat_ab1_dcx:

dat_ab2_dcx:

See Composite Attributes section

Error Messages from command execution

No messages to alarm & event handling

Attributes, Connections

- NAME
- VALID
- VAL_TYPE
- B0_VAL
- B1_VAL
- B2_VAL
-
- B31_VAL
- IW_VAL
- IL_VAL
- R_VAL

Events

- No event can be generated from the DAT object type.

Figure 3-9. Overview of the Object Type DAT.

3.5.3 Attributes

The object attributes are a subset of the corresponding attributes of the DAT residing in the ABB Master Process Station.

Table 3-32. DAT object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
VALID	IntSmall	Represents the status of the object as one bit
VAL_TYPE	IntSmall	Value type, Integer Word or Long, Boolean 32 or Float, See Table 3-31
B0_VAL	Boolean	Boolean value 0
B1_VAL	Boolean	Boolean value 1
B2_VAL	Boolean	etc.
. .	. .	
B30_VAL	Boolean	Boolean value 30
B31_VAL	Boolean	Boolean value 31
IW_VAL	IntShort	Integer word value
IL_VAL	IntLong	Integer long value
R_VAL	Float[12,4]	Real value

3.5.4 Connections

No connections exist for the DAT object type.

3.5.5 Composite Attributes

dat_ab1_dcx:

VALID, VAL_TYPE, IL_VAL

dat_ab2_dcx:

NAME, VALID, VAL_TYPE, IL_VAL

3.5.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the DAT ORDER command which is almost identical with the ORDER command. The corresponding operation parameters and the behavior of the object on such a command are also explained.

The syntax of the DAT_ORDER command is as follows:

DAT_ORDER (opCode, opProp, typeOfReq, eventLog, dialogID, dataType, aValue, ilValue, iValue, ibValue, bValue, status, textIndex)

The underlined parameters differ from the ORDER command. They have the following functionality:

Table 3-33. Parameters to the Command DAT_ORDER

Parameter	**Direction**	**OTDL- and C-Format**	**Value/explanation**
dataType	input	IntSmall - OmfSmall	Data type. See Table 3-4
ilValue	input	IntLong- OmfLong	Integer Long value. When dataType is dcINTL_DCX
iValue	input	IntShort - OmfShort	Integer word value. When dataType is dcINTW_DCX
ibValue	input	IntSmall - OmfSmall	Integer byte value. Not used.
bValue	input	Boolean - gcBoolean	Boolean value. When dataType is dcBOOL_DCX

Table 3-34. The DAT Object Type operation on the DAT_ORDER Command

Type of function for corresponding parameters	**Parameters to the DAT_ORDER command: Opcode OpProp**	**Functionality Output Exceptions**
BOOLEAN Command to change the first Boolean value	dcVALUE_CHANGE_DCX dcDAT_BOOLEAN datatype = dcBOOL_VAL_DCX	Change the B0_VAL to a new value.
WORD Command to change the Integer Word value	dcVALUE_CHANGE_DCX dcDAT_INT_WORD datatype = dcINTW_VAL_DCX	Change the I_VAL to a new value.

Table 3-34. The DAT Object Type operation on the DAT_ORDER Command (Continued)

INTEGER Command to change the Integer Long value	dcVALUE_CHANGE_DCX dcDAT_INT_LONG datatype = dcINTL_VAL_DCX	Change the IL_VAL to a new value.
REAL Command to change the real value	dcVALUE_CHANGE_DCX dcDAT_REAL datatype = dcREAL_VAL_DCX	Change the R_VAL to a new value.

3.5.7 Events

No events can be generated from the DAT object type.

3.5.8 SQL Views

Table 3-35. Description DAT

Name	Null?	Type
NAME		VARCHAR2(20)
VALID	NOT NULL	NUMBER(3)
VAL_TYPE	NOT NULL	NUMBER(3)
IW_VAL	NOT NULL	NUMBER(5)
IL_VAL	NOT NULL	NUMBER(10)
R_VAL	NOT NULL	FLOAT(24)
B0_VAL	NOT NULL	NUMBER(3)
B1_VAL	NOT NULL	NUMBER(3)
B2_VAL	NOT NULL	NUMBER(3)
B3_VAL	NOT NULL	NUMBER(3)
B4_VAL	NOT NULL	NUMBER(3)
B5_VAL	NOT NULL	NUMBER(3)
B6_VAL	NOT NULL	NUMBER(3)
B7_VAL	NOT NULL	NUMBER(3)
B8_VAL	NOT NULL	NUMBER(3)
B9_VAL	NOT NULL	NUMBER(3)
B10_VAL	NOT NULL	NUMBER(3)

Table 3-35. Description DAT (Continued)

Name	**Null?**	**Type**
B11_VAL	NOT NULL	NUMBER(3)
B12_VAL	NOT NULL	NUMBER(3)
B13_VAL	NOT NULL	NUMBER(3)
B14_VAL	NOT NULL	NUMBER(3)
B15_VAL	NOT NULL	NUMBER(3)
B16_VAL	NOT NULL	NUMBER(3)
B17_VAL	NOT NULL	NUMBER(3)
B18_VAL	NOT NULL	NUMBER(3)
B19_VAL	NOT NULL	NUMBER(3)
B20_VAL	NOT NULL	NUMBER(3)
B21_VAL	NOT NULL	NUMBER(3)
B22_VAL	NOT NULL	NUMBER(3)
B23_VAL	NOT NULL	NUMBER(3)
B24_VAL	NOT NULL	NUMBER(3)
B25_VAL	NOT NULL	NUMBER(3)
B26_VAL	NOT NULL	NUMBER(3)
B27_VAL	NOT NULL	NUMBER(3)
B28_VAL	NOT NULL	NUMBER(3)
B29_VAL	NOT NULL	NUMBER(3)
B30_VAL	NOT NULL	NUMBER(3)
B31_VAL	NOT NULL	NUMBER(3)

3.6 TEXT Object - TEXT

3.6.1 How to Use

The Object Type TEXT exists as Basic Object type in AdvaInform Object Handling and as process object type in ABB Master Process Station.

The TEXT object type is adapted for text input and output. It has specific status information and behavior to suit text dialogues. It has no event and alarm handling but a suitable set of commands. In addition to the text value it also includes storage for a floating point, an integer and a set of booleans. The object type can therefore be used in most text dialogues and text applications.

The object type Text, hereafter called TEXT, incorporates functions for:

- Storage of received input (calculated)
- Support for subscription requests (on demand, 3 cyclic intervals)
- Services to treat Commands
- Update blocking and input simulation

The TEXT object consists of a database instance for each object, for storage of inputs and various other descriptive properties and control parameters.

The behavior in MAN and AUTO mode is:

- In AUTO mode it is allowed for a program to modify the object attributes but not for an operator.
- In MAN mode it is allowed for an operator but not for a program to modify the object attributes.

The IMS Text object does not support the ord_blk attribute. The reason is that the data type bitset8 is not supported by the core system.

3.6.2 Overview

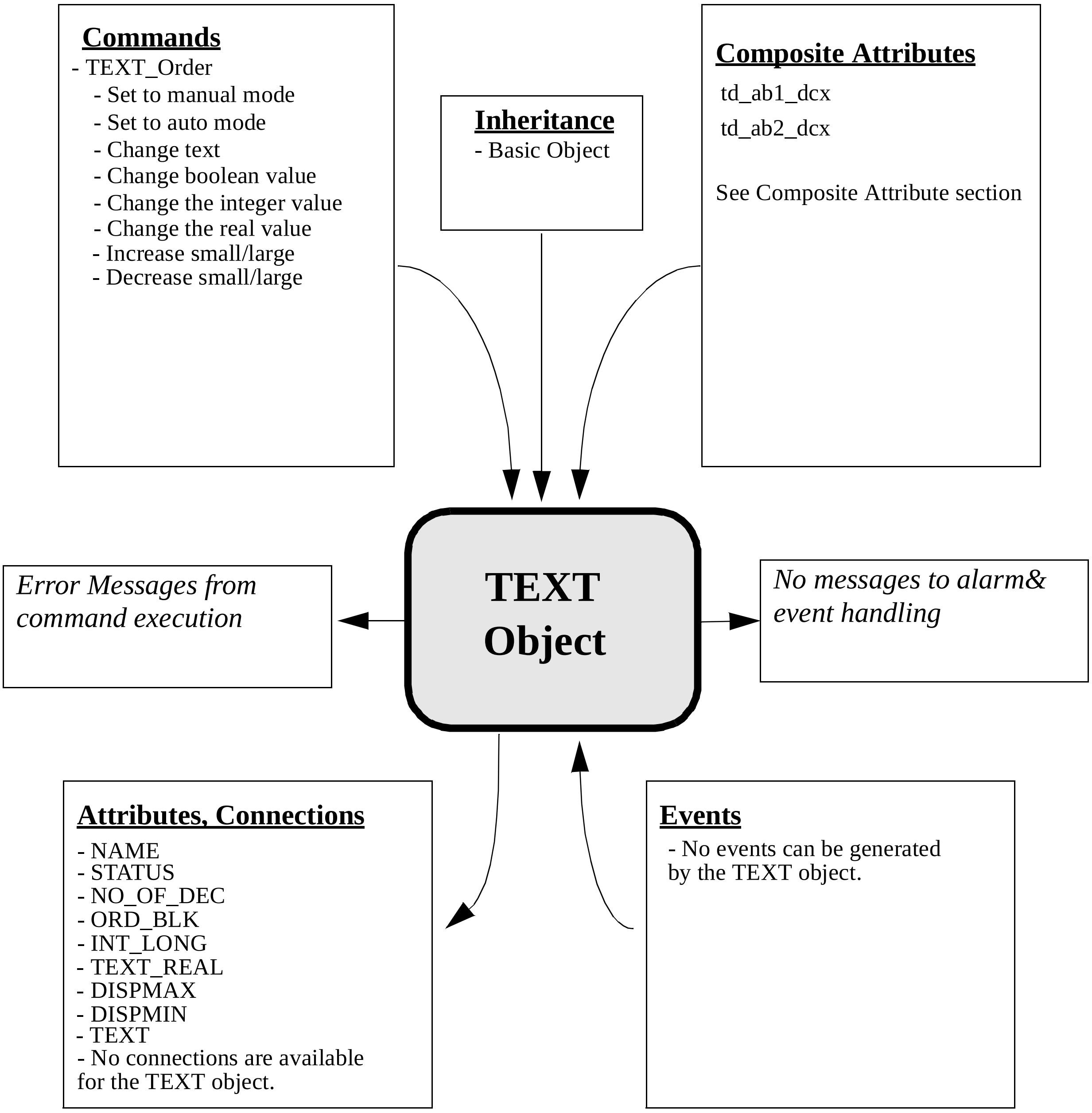


Figure 3-10. Overview of the Object Type TEXT.

3.6.3 Attributes

The object attributes are a subset of the corresponding attributes of the TEXT residing in the ABB Master Process Station.

Table 3-36. TEXT object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
STATUS	TextStatus	Represents the status of the object. See Table 3-37
NO_OF_DEC	IntSmall	No of decimals for text_real.
ORD_BLK	IntSmall	Is the blocking of manual orders changing the values. Always 1 in IMS.
INT_LONG	IntLong	Integer long value.
TEXT_REAL	Float[12,4] 16,4	Real value.
DISPMAX	Float[12,4] 16,4	Maximum value to be entered for Text_Real.
DISPMIN	Float[12,4] 16,4	Minimum value to be entered for Text_Real.
TEXT	String[73]	Text string.

Table 3-37. TdStatus. TEXT Status bits

Bits in TdStatus	Bit number	Description
VALID	1	Object is updated since restart.
SELECTED	2	Object is selected by an operator.
MAN	3	Object in manual mode, only operator updates accepted.
TBOOLEAN	4	Boolean value.
COLOUR1	5	(1)
COLOUR2	6	(1)
BLANKT	7	(1)
BLANKB	8	(1)
BLANKR	9	(1)
BLANKIL	10	(1)
B1_VAL	11	Boolean value

Table 3-37. TdStatus. TEXT Status bits (Continued)

Bits in TdStatus	Bit number	Description
B2_VAL	12	Boolean value
B3_VAL	13	Boolean value
B4_VAL	14	Boolean value

(1) Not used in IMS Basic Object

3.6.4 Connections

No connections are available for the TEXT object type.

3.6.5 Composite Attributes

td_ab1_dcx:

NO_OF_DEC, STATUS, INT_LONG, TEXT_REAL, TEXT

td_ab2_dcx:

NO_OF_DEC, STATUS, ORD_BLK, INT_LONG, TEXT_REAL, DISPMAX, DISPMIN, TEXT, NAME

3.6.6 Commands

The commands are treated in the same way as for calculated objects in ABB MasterPieces. The table below shows the TEXT ORDER command which is almost identical with the ORDER command. The corresponding operation parameters and the behavior of the object on such a command are also explained.

The syntax of the TEXT_ORDER command is as follows:

TEXT_ORDER (opCode, opProp, typeOfReq, eventLog, dialogID, dataType, aValue, dValue, bValue, tValue, status, textIndex)

The underlined parameters differ from the ORDER command. They have the following functionality:

Table 3-38. Parameters to the Command TEXT_ORDER

Parameter	Direction	OTDL- and C-Format	Value/explanation
dataType	input	IntSmall - OmfSmall	Data type. Not used. Set it to 0.
bValue	input	Boolean - gcBoolean	New boolean value.
tValue	input	String (73) - gcString	New text value.

Table 3-39. TEXT Object Type operation on the TEXT_ORDER Command

Type of function for corresponding parameters	Parameters to the TEXT_ORDER command: Opcode OpProp	Functionality Output Exceptions
MAN Set to Manual mode	dcMANUAL_DCX -	If not already manual, set the STATUS bit MAN_MODE.
AUTO Set to Automatic mode	dcAUTO_DCX -	If not already auto, reset the STATUS bit MAN_MODE.
BOOLEAN Command to change the Boolean value	dcVALUE_CHANGE_DCX dcTD_BOOL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the TBOOLEAN status bit to true.
B1_VAL Command to change the B1_VAL value	dcVALUE_CHANGE_DCX dcTD_B1_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B1_VAL status bit to true.
B2_VAL Command to change the B2_VAL value	dcVALUE_CHANGE_DCX dcTD_B2_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B2_VAL status bit to true.
B3_VAL Command to change the B3_VAL value	dcVALUE_CHANGE_DCX dcTD_B3_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B3_VAL status bit to true.
B4_VAL Command to change the B4_VAL value	dcVALUE_CHANGE_DCX dcTD_B4_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B4_VAL status bit to true
TEXT Command to change the text	dcVALUE_CHANGE_DCX dcTD_TEXT_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the TEXT to true.
INTEGER Command to change the integer value	dcVALUE_CHANGE_DCX dcTD_INTL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the INT_LONG to true.
Command to change the real value	dcVALUE_CHANGE_DCX dcTD_REAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured. Checked against DISP_MAX and DISP_MIN.

Table 3-39. TEXT Object Type operation on the TEXT_ORDER Command (Continued)

Increase Increase output with 0.5% or 5%	dcSMALL_INCREASE_ DCX or dcLARGE_INCREASE_ DCX dcTD_REAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured. Value is increased with 0.5 or 5% of total range. Checked against DISP_MAX and DISP_MIN.
Decrease Decrease output with 0.5% or 5%	dcSMALL_DECREASE_DCX or dcLARGE_DECREASE_ DCX dcTD_REAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode. Overridden by single connection input if such is configured. Value is decreased with 0.5 or 5% of total range. Checked against DISP_MAX and DISP_MIN.
MAN Reset the status bit MAN	dcRESET_DCX dcTD_MAN_DCX	Reset the STATUS bit MAN.
BOOLEAN Reset the Boolean value	dcRESET_DCX dcTD_BOOL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the TBOOLEAN status bit to false.
B1_VAL Reset the B1_VAL value	dcRESET_DCX dcTD_B1_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B1_VAL status bit to false.
B2_VAL Reset the B2_VAL value	dcRESET_DCX dcTD_B2_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B2_VAL status bit to false.
B3_VAL Reset the B3_VAL value	dcRESET_DCX dcTD_B3_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B3_VAL status bit to false.
B4_VAL Reset the B4_VAL value	dcRESET_DCX dcTD_B4_VAL_DCX	Req=Oper: Only allowed in MAN mode Req=Prog: Only allowed in AUTO mode Change the B4_VAL status bit to false

3.6.7 Events

No events can be generated by the TEXT object type.

3.6.8 SQL Views

Table 3-40. Description TEXT

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
INT_LONG	NOT NULL	NUMBER(10)
TEXT_REAL	NOT NULL	FLOAT(24)
DISPMAX	NOT NULL	FLOAT(24)
DISPMIN	NOT NULL	FLOAT(24)
NO_OF_DEC	NOT NULL	NUMBER(3)
STATUS	NOT NULL	NUMBER(5)
TEXT		VARCHAR2(72)
ORD_BLK	NOT NULL	NUMBER(3)

Table 3-41. Bit presentation ORD_BLK_STATUS, TEXT

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
ORD_00_BLK		NUMBER
ORD_01_BLK		NUMBER
ORD_02_BLK		NUMBER
ORD_03_BLK		NUMBER
ORD_04_BLK		NUMBER
ORD_05_BLK		NUMBER
ORD_06_BLK		NUMBER
ORD_07_BLK		NUMBER
BIT9		NUMBER
BIT10		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER

Table 3-41. Bit presentation ORD_BLK_STATUS, TEXT (Continued)

Name	Null?	Type
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER

Table 3-42. Bit presentation STATUS, TEXT

Name	Null?	Type
NAME		VARCHAR2(20)
VALID		NUMBER
SELECTED		NUMBER
MAN		NUMBER
TBOOLEAN		NUMBER
COLOUR1		NUMBER
COLOUR2		NUMBER
BLANKT		NUMBER
BLANKB		NUMBER
BLANKR		NUMBER
BLANKIL		NUMBER
B1_VAL		NUMBER
B2_VAL		NUMBER
B3_VAL		NUMBER
B4_VAL		NUMBER
BIT15		NUMBER
BIT16		NUMBER

3.7 Process Controller - PIDCON

3.7.1 How to Use

The object type Process Controller, here after called PIDCON, exists as process object type in ABB Master Process Stations.

The object type PIDCON incorporates functions for:

- Process control
- Ratio control
- Increase/decrease control
- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.7.2 Overview

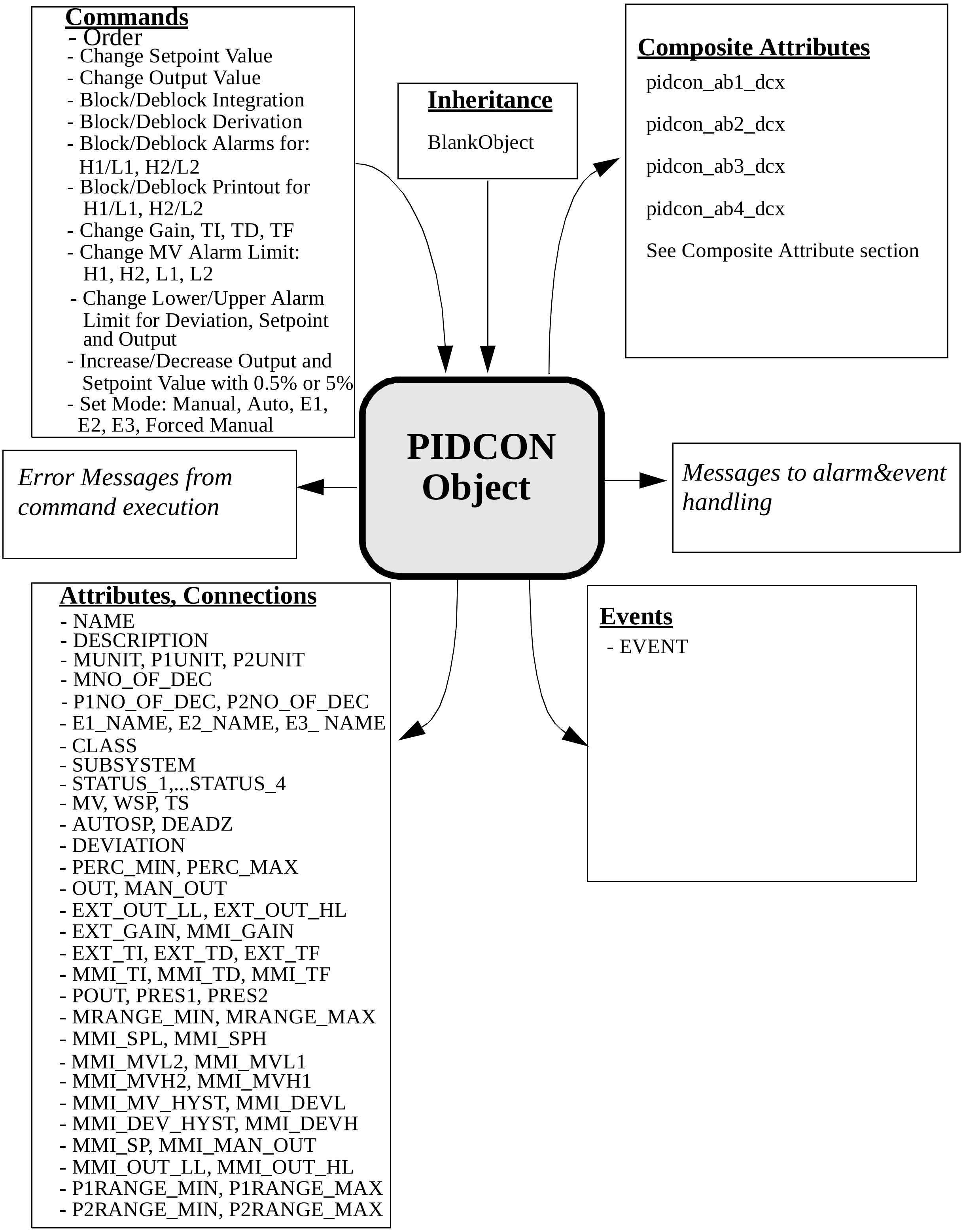


Figure 3-11. Overview of the Object Type PIDCON

3.7.3 Attributes

The object attributes are a subset of the corresponding attributes of the PIDCONs residing in the ABB Master Process Stations. Any detailed information about an attribute can be found in the ABB Master Database Element or Functional Unit manuals.

Table 3-43. PIDCON object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
MUNIT	String[7]	Measured value unit.
P1UNIT	String[7]	PRES1 unit.
P2UNIT	String[7]	PRES2 unit.
E1_NAME	TdName	Name of external reference 1.
E2_NAME	TdName	Name of external reference 2.
E3_NAME	TdName	Name of external reference 3.
MNO_OF_DEC	IntSmall	Measured value number of decimals.
P1NO_OF_DEC	IntSmall	PRES1 number of decimals.
P2NO_OF_DEC	IntSmall	PRES2 number of decimals.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Object section 0 -16.
STATUS_1	Status_1	Represents the status of the object. See Table 3-44
STATUS_2	Status_2	Represents the status of the object. See Table 3-45
STATUS_3	Status_3	Represents the status of the object. See Table 3-46
STATUS_4	Status_4	Represents the status of the object. See Table 3-47
MV	Float[8,2]	Measured value. (PARAM6)
AUTOSP	Float[8,2]	Auto setpoint. (PARAM6)
WSP	Float[8,2]	Working setpoint used when calculating the control deviation. (PARAM6)
DEVIATION	Float[8,2]	Deviation (MV-WSP). (PARAM6)
OUT	Float[8,2]	Output signal. (PARAM6)
TS	Float[8,2]	Sampling time for the controller. (PARAM6)
MAN_OUT	Float[8,2]	Manual output. (PARAM6)
EXT_OUT_LL	Float[8,2]	Low external limitation value for output. (PARAM8)
EXT_OUT_HL	Float[8,2]	High external limitation value for output. (PARAM8)
EXT_GAIN	Float[8,2]	External gain value. (PARAM9)

Table 3-43. PIDCON object type attributes (Continued)

Attribute	OTDL Data Type	Description
EXT_TI	Float[8,2]	External integration time constant. (PARAM9)
EXT_TD	Float[8,2]	External derivation time constant. (PARAM9)
EXT_TF	Float[8,2]	External filter time constant. (PARAM9)
POUT	Float[8,2]	Output to process.
PRES1	Float[8,2]	PRES1 value. Used to present actuator position.
PRES2	Float[8,2]	PRES2 value. Used to present actuator position.
MRANGE_MIN	Float[8,2]	Measured value range min.
MRANGE_MAX	Float[8,2]	Measured value range max.
MMI_SPL	Float[8,2]	Low limitation value for AUTOSP set by the operator. (PARAM7)
MMI_SPH	Float[8,2]	High limitation value for AUTOSP set by the operator. (PARAM7)
MMI_MVL2	Float[8,2]	Low alarm limitation value, L2, for MV set by the operator. (PARAM7)
MMI_MVL1	Float[8,2]	Low alarm limitation value, L1, for MV set by the operator. (PARAM7)
MMI_MVH1	Float[8,2]	High alarm limitation value, H1,for MV set by the operator. (PARAM7)
MMI_MVH2	Float[8,2]	High alarm limitation value, H2, for MV set by the operator. (PARAM7)
MMI_MV_HYST	Float[8,2]	Measured value for alarm unit hysteresis set by the operator. (PARAM7)
MMI_DEVL	Float[8,2]	Low limitation value for deviation value set by the operator.(PARAM7)
MMI_DEVH	Float[8,2]	High limitation value for deviation value set by the operator. (PARAM7)
MMI_DEV_HYST	Float[8,2]	Deviation value for alarm limit hysteresis set by the operator. (PARAM7)
MMI_SP	Float[8,2]	Setpoint value set by the operator set by the operator. (PARAM2)
MMI_MAN_OUT	Float[8,2]	Manual output value set by the operator. (PARAM2)
MMI_GAIN	Float[8,2]	Gain value set by the operator. (PARAM2)
MMI_TI	Float[8,2]	Integration time constant set by the operator. (PARAM2)
MMI_TD	Float[8,2]	Derivation time constant set by the operator. (PARAM2)
MMI_TF	Float[8,2]	Filter time constant set by the ooerator. (PARAM2)
MMI_OUT_LL	Float[8,2]	Low limitation value for output. (PARAM2)
MMI_OUT_HL	Float[8,2]	High limitation value for output. (PARAM2)
DEADZ	Float[8,2]	Deadzone
P1RANGE_MIN	Float[8,2]	PRES1 range min.
P1RANGE_MAX	Float[8,2]	PRES1 range max.
P2RANGE_MIN	Float[8,2]	PRES2 range min.

Table 3-43. PIDCON object type attributes (Continued)

Attribute	OTDL Data Type	Description
P2RANGE_MAX	Float[8,2]	PRES2 range max.
PERC_MIN	Float[8,2]	Not used.
PERC_MAX	Float[8,2]	Not used.

Table 3-44. Status_1. Status bits

Bits in status_1	Bit number	Description
IMPLEMENTED	1	Implemented
RUNNING	3	Running
ACTUATOR	4	Actuator
MVDIR	5	Measured value direction (increasing/decreasing).
SELECTED	6	Object is selected.
TESTED	8	Object is tested.
PANEL_CTRL	9	Panel control.
ON_OFF_CTRL	10	On/off control.
ALARM_UNACK	17	Alarm unacknowledged.
REP_FAIL_BLK	18	Repeats failed block.
DIST	19	Disturbance. Some kind of abnormal state exists.
ALARM_F1_BLK	20	Alarms H1/L1 are blocked.
AL_F1_PE_BLK	21	Alarms H1/L1 period block.
ALARM_F2_BLK	22	Alarms H2/L2 are blocked.
AL_F2_PE_BLK	23	Alarms H2/L2 period block.
OUT_EQ_LL	25	Output equals low limit value.
OUT_EQ_HL	26	Output equals high limit value.
SP_EQ_LL	27	Setpoint equals low limit value.
SP_EQ_HL	28	Setpoint equals high limit value.

Table 3-45. Status_2. Status bits

Bits in status_2	Bit number	Description
TS_MV_LT_L1	1	Treated fault status Measured Value less than low limit 1.
TS_MV_GT_H1	2	Treated fault status Measured Value greater than high limit 1.
TS_DEV_LT_L	3	Treated fault status Deviation less than low limit.
TS_DEV_GT_H	4	Treated fault status Deviation greater than high limit.
TS_MV_LT_L2	5	Treated fault status Measured Value less than low limit 2.
TS_MV_GT_H2	6	Treated fault status Measured Value less than high limit 2.
TS_LOCAL_FL	7	Treated fault status Local Flash.
TS_MAN_FL	8	Treated fault status Manual Flash.
TS_AUTO_FL	9	Treated fault status Auto Flash.
TS_E1_FL	10	Treated fault status E1 Flash.
TS_E2_FL	11	Treated fault status E2 Flash.
TS_E3_FL	12	Treated fault status E3 Flash.
AI_ERROR	13	AI error.
AO_ERROR	14	AO error.
DCM_ERROR	15	DCM error.
PC_BLK	16	AMPL PC block.
SERVUC	17	Service unit connected.
HW_ERR	18	Hardware error.
MMI_MANF	25	Command forced manual mode.
MMI_MAN	26	Command manual mode.
MMI_AUTO	27	Command auto mode.
MMI_INT_BLK	28	Integration block.
MMI_DER_BLK	29	Derivation block.
MMI_E1	30	Command E1 mode.
MMI_E2	31	Command E2 mode.
MMI_E3	32	Command E3 mode.

Table 3-46. Status_3. Status bits

Bits in status_3	Bit number	Description
REMOTE	1	Remote
CENTRAL	2	Central
LOCAL	3	Local
SERVICE_UNIT	4	Service unit
BAL	17	Balanced mode selected.
MAN	18	Manual mode selected.
AUTO	19	Auto mode selected.
E1	20	E1 mode selected.
E2	21	E2 mode selected.
E3	22	E3 mode selected.
LOCAL_OUT	23	Local output.
CLAMP_OUT	24	Clamp output.
EXT_OUT_LI	25	External output limit.
EXT_GAIN_ENB	26	External gain enabled.
EXT_TI_ENBL	27	External integration time enabled.
EXT_TD_ENBL	28	External derivation time enabled.
EXT_TF_ENBL	29	External filter time enabled.
EXT_INT_BLK	30	External integration block.
EXT_DER_BLK	31	External derivation block.

Table 3-47. Status_4. Status bits

Bits in status_4	Bit number	Description
PRINT_F1_BLK	3	Print F1 block.
REF1_FAIL_BLK	4	Repeat F1 fail control.
PRINT_F2_BLK	11	Print F2 block.
REF2_FAIL_BLK	12	Repeat F2 fail control.
AU_MV_LT_L1	17	Unacknowledged alarm Measured Value less than low limit 1.
AU_MV_GT_H1	18	Unacknowledged alarm Measured Value greater than high limit 1.
AU_DEV_LT_L	19	Unacknowledged alarm Deviation less than low limit.
AU_DEV_GT_H	20	Unacknowledged alarm Deviation greater than high limit.
AU_MV_LT_L2	25	Unacknowledged alarm Measured Value less than low limit 2.
AU_MV_GT_H2	26	Unacknowledged alarm Measured Value greater than high limit 2.

3.7.4 Connections

No connections are available for the PIDCON object type.

3.7.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

pidcon_ab1_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, AUTOSP, WSP, DEVIATION, OUT, TS, EXT_OUT_HL, EXT_OUT_LL, MAN_OUT, EXT_GAIN, EXT_TI, EXT_TD, EXT_TF, POUT, PRES1, PRES2, MNO_OF_DEC

pidcon_ab2_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, AUTOSP, WSP, DEVIATION, OUT, TS, EXT_OUT_HL, EXT_OUT_LL, MAN_OUT, EXT_GAIN, EXT_TI, EXT_TD, EXT_TF, POUT, PRES1, PRES2, MNO_OF_DEC, MRANGE_MAX, MRANGE_MIN, NAME, DESCRIPTION

pidcon_ab3_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, AUTOSP, WSP, DEVIATION, OUT, TS, EXT_OUT_HL, EXT_OUT_LL, MAN_OUT, EXT_GAIN, EXT_TI, EXT_TD, EXT_TF, POUT, PRES1, PRES2, MNO_OF_DEC, MRANGE_MAX, MRANGE_MIN, MMI_SPL, MMI_SPH, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_MVH2, MMI_MV_HYST, MMI_DEVL, MMI_DEVH, MMI_DEV_HYST, NAME, DESCRIPTION, MUNIT

pidcon_ab4_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, AUTOSP, WSP, DEVIATION, OUT, TS, EXT_OUT_HL, EXT_OUT_LL, MAN_OUT, EXT_GAIN, EXT_TI,

EXT_TD, EXT_TF, POUT, PRES1, PRES2, MNO_OF_DEC, MRANGE_MAX, MRANGE_MIN, MMI_SPL, MMI_SPH, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_MVH2, MMI_MV_HYST, MMI_DEVL, MMI_DEVH, MMI_DEV_HYST, MMI_SP, MMI_MAN_OUT, MMI_GAIN, MMI_TI, MMI_TD, MMI_TF, MMI_OUT_LL, MMI_OUT_HL, DEADZ, P1RANGE_MAX, P1RANGE_MIN, PERC_MIN, PERC_MAX, P1NO_OF_DEC, SUBSYSTEM, NAME, DESCRIPTION, MUNIT, P1UNIT, E1_NAME, E2_NAME, E3_NAME

3.7.6 Commands

The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-48. PIDCON operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command: Opcode OpProp**	**Functionality Output Exceptions**
SETPOINT Command for changing the setpoint value	dcVALUE_CHANGE_DCX dcSETPOINT_DCX	Changes the setpoint value.
MAN OUT Command for changing the output value	dcVALUE_CHANGE_DCX dcOUTPUT_DCX	Changes the output value.
MAN FD Set to forced manual mode	dcMAN_FORCED_DCX -	If not already in manual, the STATUS_2 bit MMI_MANF is set to true. Blocks program input updates.
MAN Set to manual mode	dcMANUAL_DCX -	If not already in manual, the STATUS_2 bit MMI_MAN is set to true. Blocks program input updates.
AUTO Set to auto mode	dcAUTO_DCX -	If not already in auto, the STATUS_2 bit MMI_AUTO is set to true.
E1 External reference E1	dcMODE_E1_DCX dcMAIN_DCX	The STATUS_2 bit MMI_E1 is set to true.
E2 External reference E2	dcMODE_E2_DCX dcMAIN_DCX	The STATUS_2 bit MMI_E2 is set to true.
E3 External reference E3	dcMODE_E3_DCX dcMAIN_DCX	The STATUS_2 bit MMI_E3 is set to true.
Integration Block Blocks integration	dcBLOCK_DCX dcINTEGRATION_DCX	If not already blocked, the STATUS_2 bit MMI_INT_BLK is set to true. Blocks integration.
Integration Deblock Deblocks integration	dcDEBLOCK_DCX dcINTEGRATION_DCX	If blocked, the STATUS_2 bit MMI_INT_BLK is set to true.
Derivation Block Blocks derivation	dcBLOCK_DCX dcDERIVATION_DCX	If not already blocked, the STATUS_2 bit MMI_DER_BLK is set to true.

Table 3-48. PIDCON operation on the ORDER Command (Continued)

Derivation Deblock Deblocks derivation	dcDEBLOCK_DCX dcDERIVATION_DCX	If blocked, the STATUS_2 bit MMI_DER_BLK is set to false.
Block H1/L1 Blocks alarms handling H1/L1	dcBLOCK_DCX dcFAULT1_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F1_BLK is set to true.
Deblock H1/L1 Deblocks alarms handling H1/L1	dcDEBLOCK_DCX dcFAULT1_ALARM_DCX	If blocked the STATUS_ bit ALARM_F1_BLK is set to false.
Block H2/L2 Blocks alarms handling H2/L2	dcBLOCK_DCX dcFAULT2_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F2_BLK is set to true.
Deblock H2/L2 Deblocks alarms handling H2/L2	dcDEBLOCK_DCX dcFAULT2_ALARM_DCX	If blocked the STATUS_ 1 bit ALARM_F2_BLK is set to false.
Print Block H1/L1 Blocks printout at alarm limit H1/L1	dcBLOCK_DCX dcFAULT1_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F1_BLK is set to true.
Print Deblock H1/L1 Deblocks printout at alarm limit H1/L1	dcDEBLOCK_DCX dcFAULT1_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F1_BLK is set to false.
Print Block H2/L2 Blocks printout at alarm limit H2/L2	dcBLOCK_DCX dcFAULT2_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F2_BLK is set to true.
Print Deblock H2/L2 Deblocks printout at alarm limit H2/L2	dcDEBLOCK_DCX dcFAULT2_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F2_BLK is set to false.
GAIN Command to change the gain value	dcVALUE_CHANGE_DCX dcGAIN_DCX	Changes the gain value.
TI Command to change the integration time constant	dcVALUE_CHANGE_DCX dcTI_DCX	Changes the integration time constant.
TD Command to change the derivation time constant	dcVALUE_CHANGE_DCX dcTD_DCX	Changes the derivation time constant.
TF Command to change the filter time	dcVALUE_CHANGE_DCX dcTF_DCX	Changes the filter time constant.
DEADZONE Command to change the deadzone	dcVALUE_CHANGE_DCX dcDEAD_ZONE_DCX	Changes the deadzone.

Table 3-48. PIDCON operation on the ORDER Command (Continued)

MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the max and min range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the max and min range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the max and min range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.
MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the max and min range. Must also be smaller than the L1 limit.
DEVH1 Command to change the upper warning limit for deviation	dcVALUE_CHANGE_DCX dcDEV_HILIM_DCX	Changes the upper warning limit for deviation.
DEVL1 Command to change the lower warning limit for deviation	dcVALUE_CHANGE_DCX dcDEV_LOLIM_DCX	Changes the lower warning limit for deviation.
SETPH Command to change the upper warning limit setpoint	dcVALUE_CHANGE_DCX dcSETP_HILIM_DCX	Changes the upper warning limit setpoint.
SETPL Command to change the lower warning limit setpoint	dcVALUE_CHANGE_DCX dcSETP_LOLIM_DCX	Changes the lower warning limit setpoint.
OUTPH Command to change the upper warning limit output	dcVALUE_CHANGE_DCX dcOUTP_HILIM_DCX	Changes the upper warning limit output.
OUTPL Command to change the lower warning limit output	dcVALUE_CHANGE_DCX dcOUTP_LOLIM_DCX	Changes the lower warning limit output.
Small Increase Increase the output value with 0.5% of range.	dcSMALL_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 0.5% of total range.

Table 3-48. PIDCON operation on the ORDER Command (Continued)

Large Increase Increase the output value with 5% of range.	dcLARGE_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 5% of total range.
Small Decrease Decrease the output value with 0.5% of range.	dcSMALL_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 0.5% of total range.
Large Decrease Decrease the output value with 5% of range.	dcLARGE_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 5% of total range.
Small Increase Increase the setpoint value with 0.5% of range.	dcSMALL_INCREASE_DCX dcSETPOINT_DCX	Value is increased with 0.5% of total range.
Large Increase Increase the setpoint value with 5% of range.	dcLARGE_INCREASE_DCX dcSETPOINT_DCX	Value is increased with 5% of total range.
Small Decrease Decrease the setpoint value with 0.5% of range.	dcSMALL_DECREASE_DCX dcSETPOINT_DCX	Value is decreased with 0.5% of total range.
Large Decrease Decrease the setpoint value with 5% of range.	dcLARGE_DECREASE_DCX dcSETPOINT_DCX	Value is decreased with 5% of total range.

3.7.7 Events

The PIDCON object type has one event called EVENT.

Table 3-49. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the PIDCON objects always request for at least the STATUS_x attributes to be returned.

3.7.8 SQL Views

Table 3-50. Description PIDCON

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)

Table 3-50. Description PIDCON (Continued)

Name	Null?	Type
MUNIT		VARCHAR2(6)
E1_NAME		VARCHAR2(20)
E2_NAME		VARCHAR2(20)
E3_NAME		VARCHAR2(20)
MNO_OF_DEC	NOT NULL	NUMBER(3)
P1NO_OF_DEC	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
SUBSYSTEM	NOT NULL	NUMBER(3)
STATUS_1	NOT NULL	NUMBER(10)
STATUS_2	NOT NULL	NUMBER(10)
STATUS_3	NOT NULL	NUMBER(10)
STATUS_4	NOT NULL	NUMBER(10)
MV	NOT NULL	FLOAT(24)
AUTOSP	NOT NULL	FLOAT(24)
WSP	NOT NULL	FLOAT(24)
DEVIATION	NOT NULL	FLOAT(24)
OUT	NOT NULL	FLOAT(24)
TS	NOT NULL	FLOAT(24)
MAN_OUT	NOT NULL	FLOAT(24)
EXT_OUT_LL	NOT NULL	FLOAT(24)
EXT_OUT_HL	NOT NULL	FLOAT(24)
EXT_GAIN	NOT NULL	FLOAT(24)
EXT_TI	NOT NULL	FLOAT(24)
EXT_TD	NOT NULL	FLOAT(24)
EXT_TF	NOT NULL	FLOAT(24)
POUT	NOT NULL	FLOAT(24)
PRES1	NOT NULL	FLOAT(24)
PRES2	NOT NULL	FLOAT(24)
MRANGE_MIN	NOT NULL	FLOAT(24)
MRANGE_MAX	NOT NULL	FLOAT(24)
MMI_SPL	NOT NULL	FLOAT(24)

Table 3-50. Description PIDCON (Continued)

Name	**Null?**	**Type**
MMI_SPH	NOT NULL	FLOAT(24)
MMI_MVL2	NOT NULL	FLOAT(24)
MMI_MVL1	NOT NULL	FLOAT(24)
MMI_MVH1	NOT NULL	FLOAT(24)
MMI_MVH2	NOT NULL	FLOAT(24)
MMI_DEVL	NOT NULL	FLOAT(24)
MMI_DEVH	NOT NULL	FLOAT(24)
MMI_SP	NOT NULL	FLOAT(24)
MMI_MAN_OUT	NOT NULL	FLOAT(24)
MMI_GAIN	NOT NULL	FLOAT(24)
MMI_TI	NOT NULL	FLOAT(24)
MMI_TD	NOT NULL	FLOAT(24)
MMI_TF	NOT NULL	FLOAT(24)
MMI_OUT_LL	NOT NULL	FLOAT(24)
MMI_OUT_HL	NOT NULL	FLOAT(24)
DEADZ	NOT NULL	FLOAT(24)
P1RANGE_MIN	NOT NULL	FLOAT(24)
P1RANGE_MAX	NOT NULL	FLOAT(24)
PERC_MIN	NOT NULL	FLOAT(24)
PERC_MAX	NOT NULL	FLOAT(24)

Table 3-51. Description STATUS_1, PIDCON

Name	Null?	Type
NAME		VARCHAR2(20)
IMPLEMENTED		NUMBER
BIT2		NUMBER
RUNNING		NUMBER
ACTUATOR		NUMBER
MVDIR		NUMBER
BIT7		NUMBER
TESTED		NUMBER
PANEL_CTRL		NUMBER
ON_OFF_CTRL		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
ALARM_UNACK		NUMBER
REP_FAIL_BLK		NUMBER
DIST		NUMBER
ALARM_F1_BLK		NUMBER
AL_F1_PE_BLK		NUMBER
ALARM_F2_BLK		NUMBER
AL_F2_PE_BLK		NUMBER
BIT24		NUMBER
OUT_EQ_LL		NUMBER
OUT_EQ_HL		NUMBER
SP_EQ_LL		NUMBER
SP_EQ_HL		NUMBER
BIT29		NUMBER
BIT30		NUMBER

Table 3-51. Description STATUS_1, PIDCON (Continued)

Name	Null?	Type
BIT31		NUMBER
BIT32		NUMBER

Table 3-52. Description STATUS_2, PIDCON

Name	Null?	Type
NAME		VARCHAR2(20)
TS_MV_LT_L1		NUMBER
TS_MV_GT_H1		NUMBER
TS_DEV_LT_L		NUMBER
TS_DEV_GT_H		NUMBER
TS_MV_LT_L2		NUMBER
TS_MV_GT_H2		NUMBER
TS_LOCAL_FL		NUMBER
TS_MAN_FL		NUMBER
TS_AUTO_FL		NUMBER
TS_E1_FL		NUMBER
TS_E2_FL		NUMBER
TS_E3_FL		NUMBER
AI_ERROR		NUMBER
AO_ERROR		NUMBER
DCM_ERROR		NUMBER
PC_BLK		NUMBER
SERVUC		NUMBER
HW_ERR		NUMBER
BIT19		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER

Table 3-52. Description STATUS_2, PIDCON (Continued)

Name	**Null?**	**Type**
MMI_MANF		NUMBER
MMI_MAN		NUMBER
MMI_AUTO		NUMBER
MMI_INT_BLK		NUMBER
MMI_DER_BLK		NUMBER
MMI_E1		NUMBER
MMI_E2		NUMBER
MMI_E3		NUMBER

Table 3-53. Description STATUS_3, PIDCON

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
REMOTE		NUMBER
CENTRAL		NUMBER
LOCAL		NUMBER
SERVICE_UNIT		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
BAL		NUMBER
MAN		NUMBER

Table 3-53. Description STATUS_3, PIDCON (Continued)

Name	Null?	Type
AUTO		NUMBER
E1		NUMBER
E2		NUMBER
E3		NUMBER
LOCAL_OUT		NUMBER
CLAMP_OUT		NUMBER
EXT_OUT_LI		NUMBER
EXT_GAIN_ENB		NUMBER
EXT_TI_ENBL		NUMBER
EXT_TD_ENBL		NUMBER
EXT_TF_ENBL		NUMBER
EXT_INT_BLK		NUMBER
EXT_DER_BLK		NUMBER
BIT32		NUMBER

Table 3-54. Description STATUS_4, PIDCON

Name	Null?	Type
NAME		VARCHAR2(20)
BIT1		NUMBER
BIT2		NUMBER
PRINT_F1_BLK		NUMBER
REF1_FAIL_BLK		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
PRINT_F2_BLK		NUMBER
REF2_FAIL_BLK		NUMBER

Table 3-54. Description STATUS_4, PIDCON (Continued)

Name	Null?	Type
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
AU_MV_LT_L1		NUMBER
AU_MV_GT_H1		NUMBER
AU_DEV_LT_L		NUMBER
AU_DEV_GT_H		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER
AU_MV_LT_L2		NUMBER
AU_MV_GT_H2		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

3.8 Ratio Station - RATIOSTN

3.8.1 How to Use

The object type Ratio Station, here after called RATIOSTN, exists as process object type in ABB Master Process Stations.

The object type RATIOSTN incorporates functions for:

- Ratio control
- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.8.2 Overview

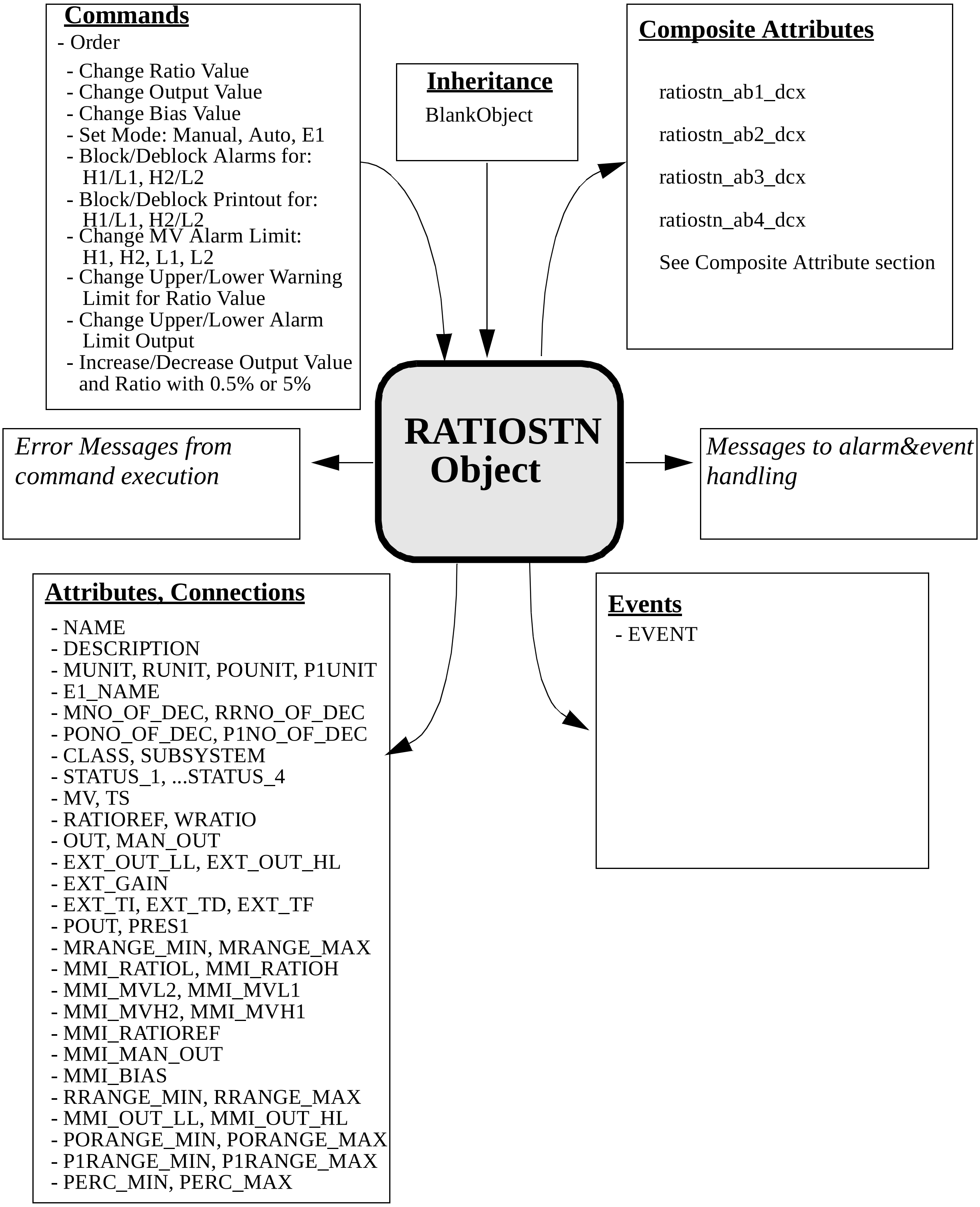


Figure 3-12. Overview of the Object Type RATIOSTN

3.8.3 Attributes

The object attributes are a subset of the corresponding attributes of the RATIOSTNs residing in the ABB Master Process Stations. Any detailed information about an attribute can be found in the ABB Master Database Element or Functional Unit manuals.

Table 3-55. RATIOSTN object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
MUNIT	String[7]	Measured value unit.
RUNIT	String[7]	Ratio unit.
POUNIT	String[7]	Process output unit
P1UNIT	String[7]	PRES1 unit.
E1_NAME	TdName	Name of external reference 1.
MNO_OF_DEC	IntSmall	Measured value number of decimals.
RRNO_OF_DEC	IntSmall	Ratio number of decimals.
PONO_OF_DEC	IntSmall	Process output number of decimals.
P1NO_OF_DEC	IntSmall	PRES1 number of decimals.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Object section 0-16.
STATUS_1	status_1	Represents the status of the object. See Table 3-56
STATUS_2	status_2	Represents the status of the object. See Table 3-57
STATUS_3	status_3	Represents the status of the object. See Table 3-58
STATUS_4	status_4	Represents the status of the object. See Table 3-59
MV	Float[8,2]	Measured value. (PARAM6)
RATIOREF	Float[8,2]	Ratio reference value. (PARAM6)
WRATIO	Float[8,2]	Working setpoint. (PARAM6)
OUT	Float[8,2]	Output signal. (PARAM6)
MAN_OUT	Float[8,2]	Manual output reference value. (PARAM6)
TS	Float[8,2]	Sampling time for the controller. (PARAM6)
EXT_OUT_LL	Float[8,2]	Low external limitation value for output. (PARAM8)
EXT_OUT_HL	Float[8,2]	High external limitation value for output. (PARAM8)
EXT_GAIN	Float[8,2]	External gain value.
EXT_TI	Float[8,2]	External integration time constant.

Table 3-55. RATIOSTN object type attributes (Continued)

Attribute	**OTDL Data Type**	**Description**
EXT_TD	Float[8,2]	External derivation time constant.
EXT_TF	Float[8,2]	External filter time constant. (PARAM9)
POUT	Float[8,2]	Output value to process.
PRES1	Float[8,2]	PRES1 value.
MRANGE_MIN	Float[8,2]	Measured value range min.
MRANGE_MAX	Float[8,2]	Measured value range max.
MMI_RATIOL	Float[8,2]	Low limitation value for ratio set by the operator. (PARAM7)
MMI_RATIOH	Float[8,2]	High limitation value for ratio set by the operator. (PARAM7)
MMI_MVL2	Float[8,2]	Low alarm limitation value, L2, for MV set by the operator. (PARAM7)
MMI_MVL1	Float[8,2]	Low alarm limitation value, L1, for MV set by the operator. (PARAM7)
MMI_MVH1	Float[8,2]	High alarm limitation value, H1, for MV set by the operator. (PARAM7)
MMI_MVH2	Float[8,2]	High alarm limitation value, H2, for MV set by the operator. (PARAM7)
MMI_RATIOREF	Float[8,2]	Ratio reference value set by the operator. (PARAM2)
MMI_MAN_OUT	Float[8,2]	Manual output value set by the operator. (PARAM2)
MMI_BIAS	Float[8,2]	Bias value set by the operator. (PARAM2)
RRANGE_MIN	Float[8,2]	Ratio factor range min.
RRANGE_MAX	Float[8,2]	Ratio factor range max.
MMI_OUT_LL	Float[8,2]	Low limitation value for output set by operator. (PARAM2)
MMI_OUT_HL	Float[8,2]	High limitation value for output set by operator. (PARAM2)
PORANGE_MIN	Float[8,2]	Output to process range max.
PORANGE_MAX	Float[8,2]	Output to process range min.
P1RANGE_MIN	Float[8,2]	PRES1 range min.
P1RANGE_MAX	Float[8,2]	PRES1 range max.
PERC_MIN	Float[8,2]	Not used.
PERC_MAX	Float[8,2]	Not used.

Table 3-56. Status_1. Status bits

Bits in status_1	Bit number	Description
IMPLEMENTED	1	Implemented
RUNNING	3	Running
SELECTED	6	Object is selected.
TESTED	8	Object is tested.
PANEL_CTRL	9	Panel control.
ALARM_UNACK	17	Alarm unacknowledged.
REP_FAIL_BLK	18	Repeats failed block.
DIST	19	Disturbance. Some kind of abnormal state exists.
ALARM_F1_BLK	20	Alarms H1/L1 are blocked.
AL_F1_PE_BLK	21	Alarms H1/L1 period block.
ALARM_F2_BLK	22	Alarms H2/L2 are blocked.
AL_F2_PE_BLK	23	Alarms H2/L2 period block.
OUT_EQ_LL	25	Output equals low limit value.
OUT_EQ_HL	26	Output equals high limit value.
RATIO_EQ_LL	27	Ratio equals low limit value.
RATIO_EQ_HL	28	Ratio equals high limit value.

Table 3-57. Status_2. Status bits

Bits in status_2	Bit number	Description
TS_MV_LT_L1	1	Treated status Measured Value less than limit 1.
TS_MV_GT_H1	2	Treated status Measured Value greater than limit 1.
TS_MV_LT_L2	3	Treated status Deviation less than low limit.
TS_MV_GT_H2	4	Treated status Deviation greater than high limit.
AI_ERROR	5	AI error.
AO_ERROR	6	AO error.
DCM_ERROR	7	DCM error.
PC_BLK	8	AMPL PC block.
SERVUC	9	Service unit connected.
HW_ERR	10	Hardware error.

Table 3-57. Status_2. Status bits (Continued)

Bits in status_2	Bit number	Description
MMI_MAN	25	Command manual mode.
MMI_AUTO	26	Command auto mode.
MMI_E1	27	Command E1 mode.

Table 3-58. Status_3. Status bits

Bits in status_3	Bit number	Description
REMOTE	1	Remote
CENTRAL	2	Central
LOCAL	3	Local
SERVICE_UNIT	4	Service unit.
TRACK_B	17	
MAN	18	Manual control mode selected.
E1	19	E1 control mode selected.

Table 3-59. Status_4. Status bits

Bits in status_4	Bit number	Description
PRINT_F1_BLK	3	Print F1 block.
REF1_FAIL_BLK	4	Repeat F1 fail control.
PRINT_F2_BLK	11	Print F2 block.
REF2_FAIL_BLK	12	Repeat F2 fail control.
AU_MV_LT_L1	17	Unacknowledged alarm Measured Value less than low limit 1.
AU_MV_GT_H1	18	Unacknowledged alarm Measured Value greater than high limit 1.
AU_MV_LT_L2	25	Unacknowledged alarm Measured Value less than low limit 2.
AU_MV_GT_H2	26	Unacknowledged alarm Measured Value greater than high limit 2.

3.8.4 Connections

No connections are available for the RATIOSTN object type.

3.8.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

ratiostn_ab1_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, RATIOREF, WRATIO, OUT, MAN_OUT, TS, EXT_OUT_HL, EXT_OUT_LL, PRES1, POUT, MNO_OF_DEC, RRNO_OF_DEC, PONO_OF_DEC, MMI_RATIOREF, MMI_MAN_OUT, MMI_BIAS, MMI_OUT_LL, MMI_OUT_HL

ratiostn_ab2_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, RATIOREF, WRATIO, OUT, MAN_OUT, TS, EXT_OUT_HL, EXT_OUT_LL, PRES1, POUT, MNO_OF_DEC, MRANGE_MAX, MRANGE_MIN, MMI_RATIOREF, MMI_MAN_OUT, MMI_BIAS, MMI_OUT_LL, MMI_OUT_HL, RRNO_OF_DEC, PONO_OF_DEC, NAME, DESCRIPTION

ratiostn_ab3_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, RATIOREF, WRATIO, OUT, MAN_OUT, TS, EXT_OUT_HL, EXT_OUT_LL, PRES1, POUT, MNO_OF_DEC, MRANGE_MAX, MRANGE_MIN, MMI_RATIOL, MMI_RATIOH, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_MVH2, MMI_RATIOREF, MMI_MAN_OUT, MMI_BIAS, MMI_OUT_LL, MMI_OUT_HL, RRNO_OF_DEC, PONO_OF_DEC, NAME, DESCRIPTION, MUNIT, RUNIT, POUNIT

ratiostn_ab4_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, RATIOREF, WRATIO, OUT, MAN_OUT, TS, EXT_OUT_HL, EXT_OUT_LL, PRES1, POUT, MNO_OF_DEC, CLASS, MRANGE_MAX, MRANGE_MIN, MMI_RATIOL, MMI_RATIOH, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_MVH2, MMI_RATIOREF, MMI_MAN_OUT, MMI_BIAS, MMI_OUT_LL, MMI_OUT_HL, RRNO_OF_DEC, PONO_OF_DEC, SUBSYSTEM, NAME, DESCRIPTION, MUNIT, RUNIT, POUNIT, E1_NAME

3.8.6 Commands

The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-60. RATIOSTN operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
RATIO Command to change ratio value	dcVALUE_CHANGE_DCX dcRATIO_DCX	Changes the ratio value.
MAN OUT Command for changing the output value	dcVALUE_CHANGE_DCX dcOUTPUT_DCX	Changes the output value.
BIAS Command to change bias value	dcVALUE_CHANGE_DCX dcBIAS_DCX	Changes the bias value.
MAN Set to manual mode	dcMANUAL_DCX -	If not already in manual, the STATUS_2 bit MMI_MAN is set to true. Blocks program input updates.
AUTO Set to auto mode	dcAUTO_DCX -	If not already in auto, the STATUS_2 bit MMI_AUTO is set to true.
E1 External reference E1	dcMODE_E1_DCX dcMAIN_DCX	The STATUS_2 bit MMI_E1 is set to true.
Block H1/L1 Blocks alarms handling H1/L1	dcBLOCK_DCX dcFAULT1_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F1_BLK is set to true.
Deblock H1/L1 Deblocks alarms handling H1/L1	dcDEBLOCK_DCX dcFAULT1_ALARM_DCX	If blocked the STATUS_ bit ALARM_F1_BLK is set to false.
Block H2/L2 Blocks alarms handling H2/L2	dcBLOCK_DCX dcFAULT2_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F2_BLK is set to true.
Deblock H2/L2 Deblocks alarms handling H2/L2	dcDEBLOCK_DCX dcFAULT2_ALARM_DCX	If blocked the STATUS_ bit ALARM_F2_BLK is set to false.
Print Block H1/L1 Blocks printout at alarm limit H1/L1	dcBLOCK_DCX dcFAULT1_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F1_BLK is set to true.
Print Deblock H1/L1 Deblocks printout at alarm limit H1/L1	dcDEBLOCK_DCX dcFAULT1_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F1_BLK is set to false.

Table 3-60. RATIOSTN operation on the ORDER Command (Continued)

Print Block H2/L2 Blocks printout at alarm limit H2/L2	dcBLOCK_DCX dcFAULT2_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F2_BLK is set to true.
Print Deblock H2/L2 Deblocks printout at alarm limit H2/L2	dcDEBLOCK_DCX dcFAULT2_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F2_BLK is set to false.
MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the max and min range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the max and min range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the max and min range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.
MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the max and min range. Must also be smaller than L1 limit.
RATIOH Command to change upper warning limit ratio value	dcVALUE_CHANGE_DCX dcRATIO_HILIM_DCX	Changes the upper warning limit ratio value.
RATIOL Command to change lower warning limit ratio value	dcVALUE_CHANGE_DCX dcRATIO_LOLIM_DCX	Changes the lower warning limit ratio value.
OUTPH Command to change the upper warning limit output	dcVALUE_CHANGE_DCX dcOUTP_HILIM_DCX	Changes the upper warning limit output.
OUTPL Command to change the lower warning limit output	dcVALUE_CHANGE_DCX dcOUTP_LOLIM_DCX	Changes the lower warning limit output.
Small Increase Increase the output value with 0.5% of range	dcSMALL_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 0.5% of total range.
Large Increase Increase the output value with 5% of range	dcLARGE_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 5% of total range.
Small Decrease Decrease the output value with 0.5% of range	dcSMALL_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 0.5% of total range.

Table 3-60. RATIOSTN operation on the ORDER Command (Continued)

Large Decrease Decrease the output value with 5% of range	dcLARGE_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 5% of total range.
Small Increase Increase the ratio with 0.5% of range	dcSMALL_INCREASE_DCX dcRATIO_DCX	Ratio is increased with 0.5% of total range.
Large Increase Increase the ratio with 5% of range	dcLARGE_INCREASE_DCX dcRATIO_DCX	Ratio is increased with 5% of total range.
Small Decrease Decrease the ratio with 0.5% of range	dcSMALL_DECREASE_DCX dcRATIO_DCX	Ratio is decreased with 0.5% of total range.
Large Decrease Decrease the ratio with 5% of range	dcLARGE_DECREASE_DCX dcRATIO_DCX	Ratio is decreased with 5% of total range.

3.8.7 Events

The RATIOSTN object type has one event called EVENT:

Table 3-61. The event EVENT

Name	**Value/explanation**
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the RATIOSTN objects always request for at least the STATUS_x attributes to be returned.

3.8.8 SQL Views

Table 3-62. Description RATIOSTN

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
MNO_OF_DEC	NOT NULL	NUMBER(3)
RRNO_OF_DEC	NOT NULL	NUMBER(3)
PONO_OF_DEC	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
SUBSYSTEM	NOT NULL	NUMBER(3)
STATUS_1	NOT NULL	NUMBER(10)
STATUS_2	NOT NULL	NUMBER(10)
STATUS_3	NOT NULL	NUMBER(10)
STATUS_4	NOT NULL	NUMBER(10)
MV	NOT NULL	FLOAT(24)
RATIOREF	NOT NULL	FLOAT(24)
WRATIO	NOT NULL	FLOAT(24)
OUT	NOT NULL	FLOAT(24)
MAN_OUT	NOT NULL	FLOAT(24)
TS	NOT NULL	FLOAT(24)
EXT_OUT_LL	NOT NULL	FLOAT(24)
EXT_OUT_HL	NOT NULL	FLOAT(24)
POUT	NOT NULL	FLOAT(24)
PRES1	NOT NULL	FLOAT(24)
MRANGE_MIN	NOT NULL	FLOAT(24)
MRANGE_MAX	NOT NULL	FLOAT(24)
MMI_RATIOL	NOT NULL	FLOAT(24)
MMI_RATIOH	NOT NULL	FLOAT(24)
MMI_MVL2	NOT NULL	FLOAT(24)
MMI_MVL1	NOT NULL	FLOAT(24)
MMI_MVH1	NOT NULL	FLOAT(24)
MMI_MVH2	NOT NULL	FLOAT(24)
MMI_RATIOREF	NOT NULL	FLOAT(24)

Table 3-62. Description RATIOSTN (Continued)

Name	Null?	Type
MMI_MAN_OUT	NOT NULL	FLOAT(24)
MMI_BIAS	NOT NULL	FLOAT(24)
MMI_OUT_LL	NOT NULL	FLOAT(24)
MMI_OUT_HL	NOT NULL	FLOAT(24)
MUNIT		VARCHAR2(6)
RUNIT		VARCHAR2(6)
POUNIT		VARCHAR2(6)
E1_NAME		VARCHAR2(20)

Table 3-63. Bit presentation STATUS_1, RATIOSTN

Name	Null?	Type
NAME		VARCHAR2(20)
IMPLEMENTED		NUMBER
BIT2		NUMBER
RUNNING		NUMBER
ACTUATOR		NUMBER
BIT5		NUMBER
SELECTED		NUMBER
BIT7		NUMBER
TESTED		NUMBER
PANEL_CTRL		NUMBER
BIT10		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
ALARM_UNACK		NUMBER
REP_FAIL_BLK		NUMBER

Table 3-63. Bit presentation STATUS_1, RATIOSTN (Continued)

Name	Null?	Type
DIST		NUMBER
ALARM_F1_BLK		NUMBER
AL_F1_PE_BLK		NUMBER
ALARM_F2_BLK		NUMBER
AL_F2_PE_BLK		NUMBER
BIT24		NUMBER
OUT_EQ_LL		NUMBER
OUT_EQ_HL		NUMBER
RATIO_EQ_LL		NUMBER
RATIO_EQ_HL		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-64. Bit presentation STATUS_2, RATIOSTN

Name	Null?	Type
NAME		VARCHAR2(20)
TS_MV_LT_L1		NUMBER
TS_MV_GT_H1		NUMBER
TS_MV_LT_L2		NUMBER
TS_MV_GT_H2		NUMBER
AI_ERROR		NUMBER
AO_ERROR		NUMBER
DCM_ERROR		NUMBER
PC_BLK		NUMBER
SERVUC		NUMBER
HW_ERR		NUMBER
BIT11		NUMBER
BIT12		NUMBER

Table 3-64. Bit presentation STATUS_2, RATIOSTN (Continued)

Name	Null?	Type
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
BIT17		NUMBER
BIT18		NUMBER
BIT19		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER
MMI_MAN		NUMBER
MMI_AUTO		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-65. Bit presentation STATUS_3, RATIOSTN

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
REMOTE		NUMBER
CENTRAL		NUMBER
LOCAL		NUMBER
SERVICE_UNIT		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
TRACK_B		NUMBER
MAN		NUMBER
AUTO		NUMBER
E1		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER
BIT25		NUMBER
BIT26		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER

Table 3-65. Bit presentation STATUS_3, RATIOSTN (Continued)

Name	Null?	Type
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-66. Bit presentation STATUS_4, RATIOSTN

Name	Null?	Type
NAME		VARCHAR2(20)
BIT1		NUMBER
BIT2		NUMBER
PRINT_F1_BLK		NUMBER
REF1_FAIL_BLK		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
PRINT_F2_BLK		NUMBER
REF2_FAIL_BLK		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
AU_MV_LT_L1		NUMBER
AU_MV_GT_H1		NUMBER
BIT19		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER

Table 3-66. Bit presentation STATUS_4, RATIOSTN (Continued)

Name	**Null?**	**Type**
BIT24		NUMBER
AU_MV_LT_L2		NUMBER
AU_MV_GT_H2		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

3.9 Manual Station - MANSTN

3.9.1 How to Use

The object type Manual Station, here after called MANSTN, exists as process object type in ABB Master Process Stations.

The object type MANSTN incorporates functions for:

- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.9.2 Overview

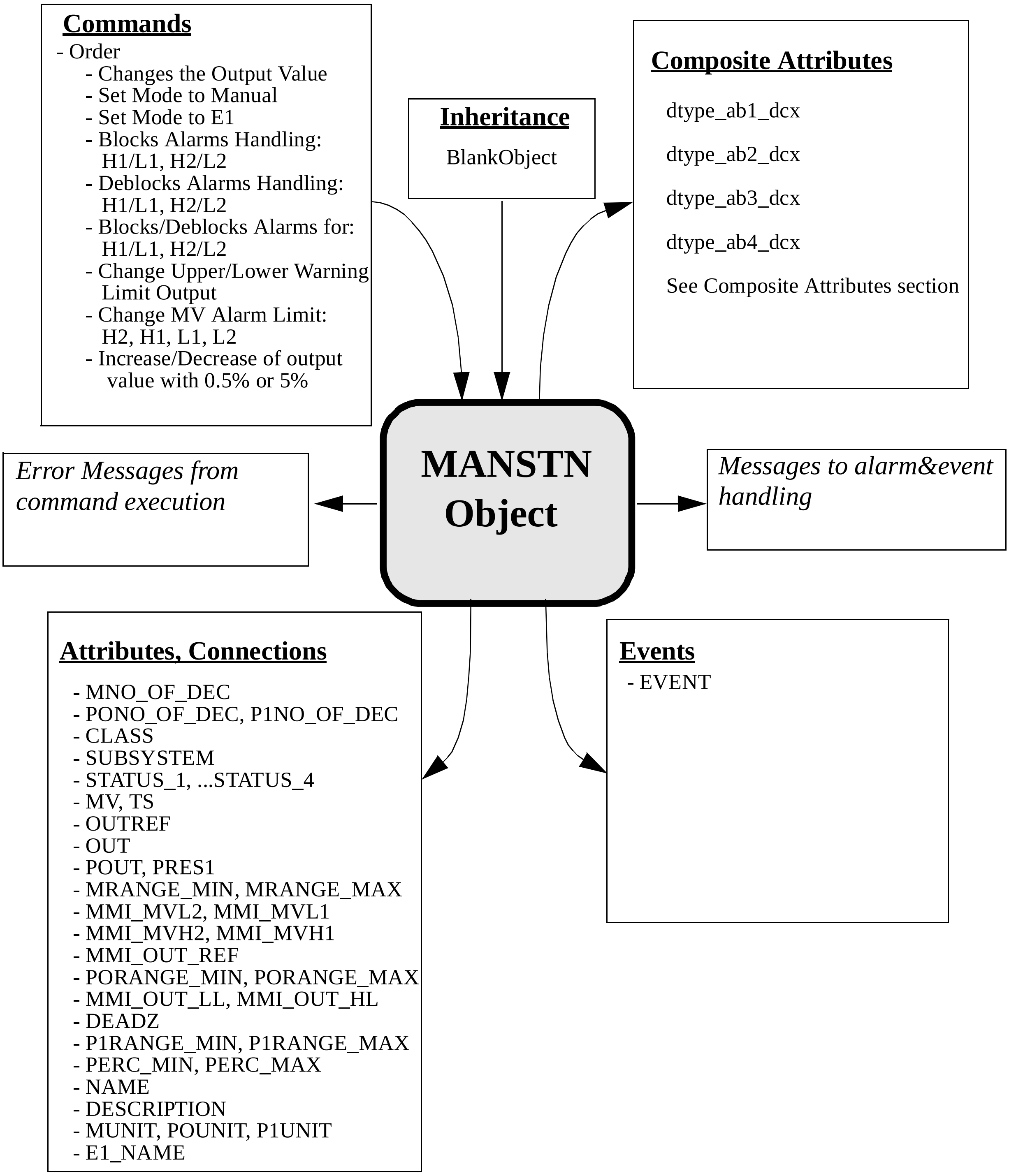


Figure 3-13. Overview of the Object Type MANSTN

3.9.3 Attributes

The object attributes are a subset of the corresponding attributes of the MANSTNs residing in the ABB Master Process Stations. Any detailed information about an attribute can be found in the ABB Master Database Element or Functional Unit manuals.

Table 3-67. MANSTN object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
MUNIT	String[7]	Measured value unit.
POUNIT	String[7]	Output to process unit.
P1UNIT	String[7]	PRES1 unit.
E1_NAME	TdName	Name of external reference 1.
MNO_OF_DEC	IntSmall	Measured value number of decimals.
PONO_OF_DEC	IntSmall	Output to process number of decimals.
P1NO_OF_DEC	IntSmall	PRES1 number of decimals.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Object section 0 -16.
STATUS_1	status_1	Represents the status of the object. See Table 3-68
STATUS_2	status_2	Represents the status of the object. See Table 3-69
STATUS_3	status_3	Represents the status of the object. See Table 3-70
STATUS_4	status_4	Represents the status of the object. See Table 3-71
MV	FLOAT[8,2]	Measured value. (PARAM6)
OUTREF	FLOAT[8,2]	Output reference value. (PARAM 6)
OUT	FLOAT[8,2]	Output signal. (PARAM6)
TS	FLOAT[8,2]	Sampling time for the controller. (PARAM6)
POUT	FLOAT[8,2]	Output to process.
PRES1	FLOAT[8,2]	PRES1 value.
MRANGE_MIN	FLOAT[8,2]	Measured value range min.
MRANGE_MAX	FLOAT[8,2]	Measured value range max.
MMI_MVL2	FLOAT[8,2]	Low alarm limitation value, L2, for MV set by the operator. (PARAM7)
MMI_MVL1	FLOAT[8,2]	Low alarm limitation value, L1, for MV set by the operator. (PARAM7)
MMI_MVH1	FLOAT[8,2]	High alarm limitation value, H1,for MV set by the operator. (PARAM7)
MMI_MVH2	FLOAT[8,2]	High alarm limitation value, H2, for MV set by the operator. (PARAM7)

Table 3-67. MANSTN object type attributes (Continued)

Attribute	OTDL Data Type	Description
MMI_OUT_REF	FLOAT[8,2]	Manual output reference value. (PARAM 2)
PORANGE_MIN	FLOAT[8,2]	Output to process range min.
PORANGE_MAX	FLOAT[8,2]	Output to process range max.
MMI_OUT_LL	FLOAT[8,2]	Low limitation value for output. (PARAM2)
MMI_OUT_HL	FLOAT[8,2]	High limitation value for output. (PARAM2)
DEADZ	FLOAT[8,2]	Deadzone
P1RANGE_MIN	FLOAT[8,2]	PRES1 range min.
P1RANGE_MAX	FLOAT[8,2]	PRES1 range max.
PERC_MIN	FLOAT[8,2]	Not used.
PERC_MAX	FLOAT[8,2]	Not used.

Table 3-68. Status_1. Status bits

Bits in status_1	Bit number	Description
IMPLEMENTED	1	Implemented
RUNNING	3	Running
ACTUATOR	4	Actuator
MVDIR	5	Measured value direction (increasing/decreasing).
SELECTED	6	Object is selected.
TESTED	8	Object is tested.
PANEL_CTRL	9	Panel control.
ON_OFF_CTRL	10	On/off control.
ALARM_UNACK	17	Alarm unacknowledged.
REP_FAIL_BLK	18	Repeats failed block.
DIST	19	Some kind of abnormal state exists.
ALARM_F1_BLK	20	Alarms H1/L1 are blocked.
AL_F1_PE_BLK	21	Alarms H1/L1 period block.
ALARM_F2_BLK	22	Alarms H2/L2 are blocked.
AL_F2_PE_BLK	23	Alarms H2/L2 period block.
OUT_EQ_LL	25	Output equals low limit value.

Table 3-68. Status_1. Status bits (Continued)

Bits in status_1	Bit number	Description
OUT_EQ_HL	26	Output equals high limit value.
SP_EQ_LL	27	Setpoint equals low limit value.
SP_EQ_HL	28	Setpoint equals high limit value.

Table 3-69. Status_2. Status bits

Bits in status_2	Bit number	Description
TS_MV_LT_L1	1	Treated status Measured Value less than low limit 1.
TS_MV_GT_H1	2	Treated status Measured Value greater than high limit 1.
TS_MV_LT_L2	3	Treated status Measured Value less than low limit 2.
TS_MV_GT_H2	4	Treated status Measured Value greater than high limit 2.
AI_ERROR	5	AI error.
AO_ERROR	6	AO error.
DCM_ERROR	7	DCM error.
PC_BLK	8	AMPL PC block.
SERVUC	9	Service unit connected.
HW_ERR	10	Hardware error.
MMI_MAN	25	Command manual mode.
MMI_E1	26	Command E1 mode.

Table 3-70. Status_3. Status bits

Bits in status_3	Bit number	Description
REMOTE	1	Remote
CENTRAL	2	Central
LOCAL	3	Local
SERVICE_UNIT	4	Service unit
TRACK_B	17	
MAN	18	Manual control mode selected.
E1	19	E1 control mode selected.

Table 3-71. Status_4. Status bits

Bits in status_4	Bit number	Description
PRINT_F1_BLK	3	Print F1 block.
REF1_FAIL_BLK	4	Repeat F1 fail control.
PRINT_F2_BLK	11	Print F2 block.
REF2_FAIL_BLK	12	Repeat F2 fail control.
AU_MV_LT_L1	17	Unacknowledged alarm Measured Value less than low limit 1.
AU_MV_GT_H1	18	Unacknowledged alarm Measured Value greater than high limit 1.
AU_MV_LT_L2	25	Unacknowledged alarm Measured Value less than low limit 2.
AU_MV_GT_H2	26	Unacknowledged alarm Measured Value greater than high limit 2.

3.9.4 Connections

No connections are available for the MANSTN object type.

3.9.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

dtype_ab1_dcx:

> STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, OUTREF, OUT, TS, POUT, PRES1, MNO_OF_DEC, PONO_OF_DEC

dtype_ab2_dcx:

STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, OUTREF, OUT, TS, POUT, PRES1, MNO_OF_DEC, MRANGE_MIN, MRANGE_MAX, PONO_OF_DEC, NAME, DESCRIPTION

dtype_ab3_dcx:

STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, OUTREF, OUT, TS, POUT, PRES1, MNO_OF_DEC, MRANGE_MIN, MRANGE_MAX, PONO_OF_DEC, PORANGE_MIN, PORANGE_MAX, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_MVH2, P1RANGE_MIN, P1RANGE_MAX, P1NO_OF_DEC, NAME, DESCRIPTION, MUNIT, POUNIT, P1UNIT

dtype_ab4_dcx:

STATUS_1, STATUS_2, STATUS_3, STATUS_4, MV, OUTREF, OUT, TS, POUT, PRES1, MNO_OF_DEC, CLASS, MRANGE_MAX, MRANGE_MIN, PONO_OF_DEC, PORANGE_MIN, PORANGE_MAX, MMI_MVL2, MMI_MVL1, MMI_MVH1, MMI_OUT_REF, MMI_OUT_LL, MMI_OUT_HL, MMI_MVH2, P1RANGE_MIN, P1RANGE_MAX, P1NO_OF_DEC, PERC_MIN, PERC_MAX, SUBSYSTEM, NAME, DESCRIPTION, MUNIT, POUNIT, P1UNIT, E1_NAME

3.9.6 Commands

The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-72. MANSTN operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
MAN OUT Command for changing the output value	dcVALUE_CHANGE_DCX dcOUTPUT_DCX	Changes the output value.
MAN Set to manual mode	dcMANUAL_DCX -	If not already manual, the STATUS_2 bit MMI_MAN is set to true. Blocks program input updates.
E1 External reference E1	dcMODE_E1_DCX dcMAIN_DCX	The STATUS_2 bit MMI_E1 is set to true.
Block H1/L1 Blocks alarms handling H1/L1	dcBLOCK_DCX dcFAULT1_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F1_BLK is set to true.
Deblock H1/L1 Deblocks alarms handling H1/L1	dcDEBLOCK_DCX dcFAULT1_ALARM_DCX	If blocked the STATUS_ bit ALARM_F1_BLK is set to false.
Block H2/L2 Blocks alarms handling H2/L2	dcBLOCK_DCX dcFAULT2_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_F2_BLK is set to true.

Table 3-72. MANSTN operation on the ORDER Command (Continued)

Deblock H2/L2 Deblocks alarms handling H2/L2	dcDEBLOCK_DCX dcFAULT2_ALARM_DCX	If blocked the STATUS_ bit ALARM_F2_BLK is set to false.
Print Block H1/L1 Blocks printout at alarm limit H1/L1	dcBLOCK_DCX dcFAULT1_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F1_BLK is set to true.
Print Deblock H1/L1 Deblocks printout at alarm limit H1/L1	dcDEBLOCK_DCX dcFAULT1_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F1_BLK is set to false.
Print Block H2/L2 Blocks printout at alarm limit H2/L2	dcBLOCK_DCX dcFAULT2_PRINT_DCX	If not already blocked, the STATUS_4 bit PRINT_F2_BLK is set to true.
Print Deblock H2/L2 Deblocks printout at alarm limit H2/L2	dcDEBLOCK_DCX dcFAULT2_PRINT_DCX	If blocked, the STATUS_4 bit PRINT_F2_BLK is set to false.
OUTPH Command to change the upper warning limit output	dcVALUE_CHANGE_DCX dcOUTP_HILIM_DCX	Changes the upper warning limit output.
OUTPL Command to change the lower warning limit output	dcVALUE_CHANGE_DCX dcOUTP_LOLIM_DCX	Changes the lower warning limit output.
MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the min and max range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the min and max range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the min and max range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.
MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the min and max range. Must also be smaller than the L1 limit.
Small Increase Increase the output value with 0.5% of range	dcSMALL_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 0.5% of total range.
Large Increase Increase the output value with 5% of range	dcLARGE_INCREASE_DCX dcOUTPUT_DCX	Value is increased with 5% of total range.

Table 3-72. MANSTN operation on the ORDER Command (Continued)

Small Decrease Decrease the output value with 0.5% of rang.	dcSMALL_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 0.5% of total range.
Large Decrease Decrease the output value with 5% of range	dcLARGE_DECREASE_DCX dcOUTPUT_DCX	Value is decreased with 5% of total range.

3.9.7 Events

The MANSTN object type has one event called EVENT:

Table 3-73. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the MANSTN objects always request for at least the STATUS_x attributes to be returned.

3.9.8 SQL Views

Table 3-74. Description MANSTN

Name	Null?	Type
NAME		VARCHAR2(20)
DESCRIPTION		VARCHAR2(28)
MNO_OF_DEC	NOT NULL	NUMBER(3)
PONO_OF_DEC	NOT NULL	NUMBER(3)
P1NO_OF_DEC	NOT NULL	NUMBER(3)
CLASS	NOT NULL	NUMBER(3)
SUBSYSTEM	NOT NULL	NUMBER(3)
STATUS_1	NOT NULL	NUMBER(10)
STATUS_2	NOT NULL	NUMBER(10)
STATUS_3	NOT NULL	NUMBER(10)
STATUS_4	NOT NULL	NUMBER(10)
MV	NOT NULL	FLOAT(24)
OUTREF	NOT NULL	FLOAT(24)
OUT	NOT NULL	FLOAT(24)
TS	NOT NULL	FLOAT(24)
POUT	NOT NULL	FLOAT(24)
PRES1	NOT NULL	FLOAT(24)
MRANGE_MIN	NOT NULL	FLOAT(24)
MRANGE_MAX	NOT NULL	FLOAT(24)
MMI_MVL2	NOT NULL	FLOAT(24)
MMI_MVL1	NOT NULL	FLOAT(24)
MMI_MVH1	NOT NULL	FLOAT(24)
MMI_MVH2	NOT NULL	FLOAT(24)
MMI_OUT_REF	NOT NULL	FLOAT(24)
PORANGE_MIN	NOT NULL	FLOAT(24)
PORANGE_MAX	NOT NULL	FLOAT(24)
MMI_OUT_LL	NOT NULL	FLOAT(24)
MMI_OUT_HL	NOT NULL	FLOAT(24)
P1RANGE_MIN	NOT NULL	FLOAT(24)
P1RANGE_MAX	NOT NULL	FLOAT(24)

Table 3-74. Description MANSTN (Continued)

Name	Null?	Type
PERC_MIN	NOT NULL	FLOAT(24)
PERC_MAX	NOT NULL	FLOAT(24)
MUNIT		VARCHAR2(6)
POUNIT		VARCHAR2(6)
P1UNIT		VARCHAR2(6)
E1_NAME		VARCHAR2(20)

Table 3-75. Bitpresentation STATUS_1, MANSTN

Name	Null?	Type
NAME		VARCHAR2(20)
IMPLEMENTED		NUMBER
SPARE_STATUS_01		NUMBER
RUNNING		NUMBER
ACTUATOR		NUMBER
MVDIR		NUMBER
SELECTED		NUMBER
SPARE_STATUS_06		NUMBER
TESTED		NUMBER
PANEL_CTRL		NUMBER
ON_OFF_CTRL		NUMBER
SPARE_STATUS_10		NUMBER
SPARE_STATUS_11		NUMBER
SPARE_STATUS_12		NUMBER
SPARE_STATUS_13		NUMBER
SPARE_STATUS_14		NUMBER
SPARE_STATUS_15		NUMBER
ALARM_UNACK		NUMBER
REP_FAIL_BLK		NUMBER
DIST		NUMBER
ALARM_F1_BLK		NUMBER

Table 3-75. Bitpresentation STATUS_1, MANSTN (Continued)

Name	Null?	Type
AL_F1_PE_BLK		NUMBER
ALARM_F2_BLK		NUMBER
AL_F2_PE_BLK		NUMBER
BIT24		NUMBER
OUT_EQ_LL		NUMBER
OUT_EQ_HL		NUMBER
SP_EQ_LL		NUMBER
SP_EQ_HL		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-76. Bitpresentation STATUS_2, MANSTN

Name	Null?	Type
NAME		VARCHAR2(20)
TS_MV_LT_L1		NUMBER
TS_MV_GT_H1		NUMBER
TS_MV_LT_L2		NUMBER
TS_MV_GT_H2		NUMBER
AI_ERROR		NUMBER
AO_ERROR		NUMBER
DCM_ERROR		NUMBER
PC_BLK		NUMBER
SERVUC		NUMBER
HW_ERR		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER

Table 3-76. Bitpresentation STATUS_2, MANSTN (Continued)

Name	Null?	Type
BIT15		NUMBER
BIT16		NUMBER
BIT17		NUMBER
BIT18		NUMBER
BIT19		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER
MMI_MAN		NUMBER
MMI_E1		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-77. Bitpresentation STATUS_3, MANSTN

Name	**Null?**	**Type**
NAME		VARCHAR2(20)
REMOTE		NUMBER
CENTRAL		NUMBER
LOCAL		NUMBER
SERVICE_UNIT		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
BIT11		NUMBER
BIT12		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
TRACK_B		NUMBER
MAN		NUMBER
E1		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER
BIT24		NUMBER
BIT25		NUMBER
BIT26		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER

Table 3-77. Bitpresentation STATUS_3, MANSTN (Continued)

Name	Null?	Type
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

Table 3-78. Bitpresentation STATUS_4, MANSTN

Name	Null?	Type
NAME		VARCHAR2(20)
BIT1		NUMBER
BIT2		NUMBER
PRINT_F1_BLK		NUMBER
REF1_FAIL_BLK		NUMBER
BIT5		NUMBER
BIT6		NUMBER
BIT7		NUMBER
BIT8		NUMBER
BIT9		NUMBER
BIT10		NUMBER
PRINT_F2_BLK		NUMBER
REF2_FAIL_BLK		NUMBER
BIT13		NUMBER
BIT14		NUMBER
BIT15		NUMBER
BIT16		NUMBER
AU_MV_LT_L1		NUMBER
AU_MV_GT_H1		NUMBER
BIT19		NUMBER
BIT20		NUMBER
BIT21		NUMBER
BIT22		NUMBER
BIT23		NUMBER

Table 3-78. Bitpresentation STATUS_4, MANSTN (Continued)

Name	Null?	Type
BIT24		NUMBER
AU_MV_LT_L2		NUMBER
AU_MV_GT_H2		NUMBER
BIT27		NUMBER
BIT28		NUMBER
BIT29		NUMBER
BIT30		NUMBER
BIT31		NUMBER
BIT32		NUMBER

3.10 General Binary Object - GENBIN

3.10.1 How to Use

The Object Type General Binary Object, here after called GENBIN, exists as process object type in ABB Master Process Station.

The object type GENBIN incorporates functions for:

- User defined control function
- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.10.2 Overview

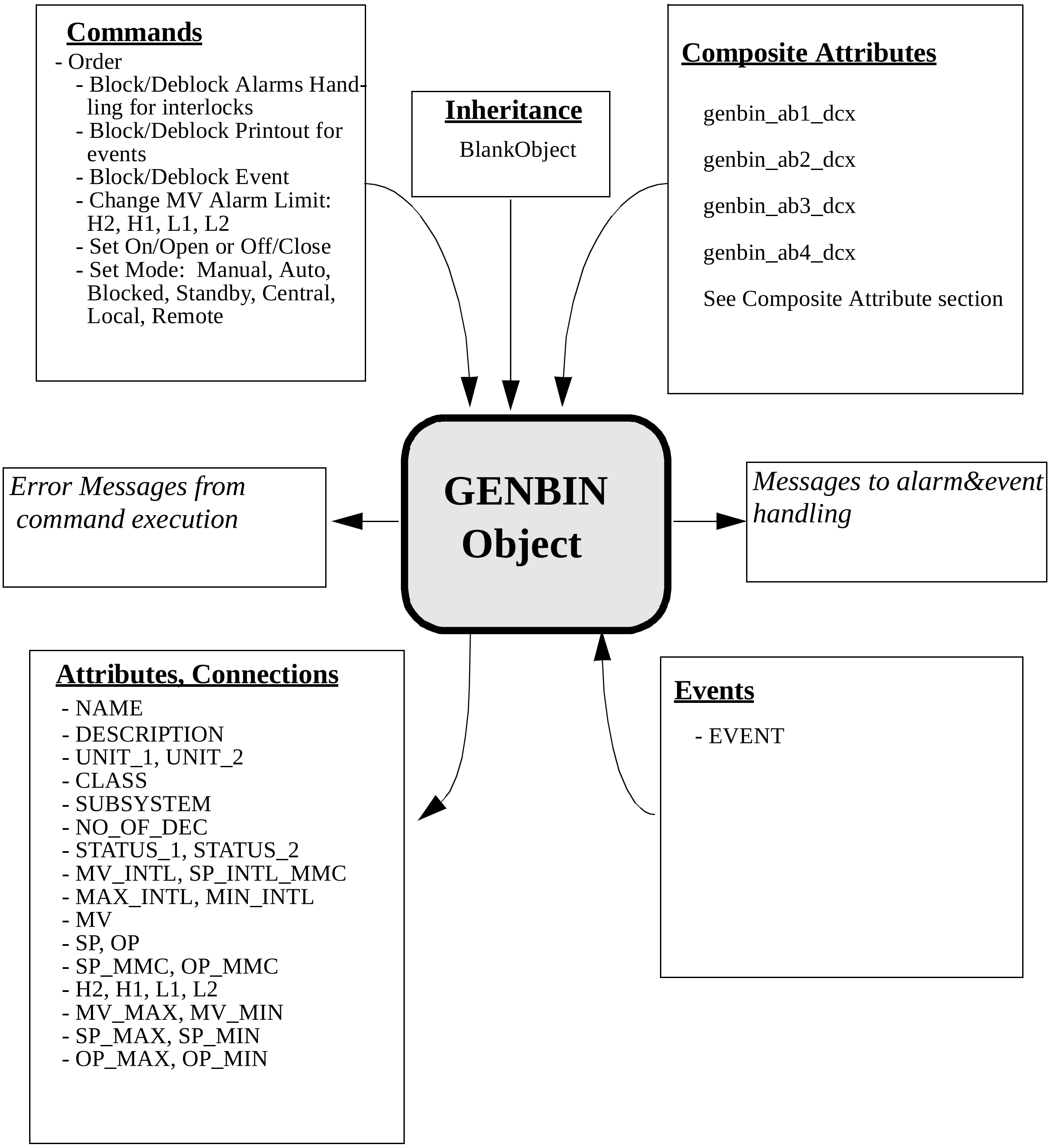


Figure 3-14. Overview of the Object Type GENBIN

3.10.3 Attributes

The object attributes are a subset of the corresponding attributes of the GENBINs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master Database Element or Functional Units manuals.

Table 3-79. GENBIN object type attributes.

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
UNIT_1	String[7]	The unit of the measured value in percentage in engineering unit.
UNIT_2	String[7]	The unit of the measured value 2 and setpoint 2, in percentage, in engineering unit.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Process section 0 -16.
NO_OF_DEC	IntSmall	Number of decimals.
STATUS_1	TdStatus	Represents the status of the object. See Table 3-80
STATUS_2	Status2	Represents the status of the object. See Table 3-81
MV_INTL	IntLong	Measured integer value.
SP_INTL_MMC	IntLong	Setpoint integer value.
MAX_INTL	IntLong	Maximum integer value.
MIN_INTL	IntLong	Minimum integer value.
MV	Float[8,2]	Measured value.
SP	Float[8,2]	Setpoint value.
OP	Float[8,2]	Output value.
SP_MMC	Float[8,2]	Preferred setpoint value set by the operator.
OP_MMC	Float[8,2]	Preferred output value set by the operator.
H2	Float[8,2]	Upper alarm limit H2.
H1	Float[8,2]	Upper alarm limit H1.
L1	Float[8,2]	Lower alarm limit L1.
L2	Float[8,2]	Lower alarm limit L2.
MV_MAX	Float[8,2]	Maximum measured value.
MV_MIN	Float[8,2]	Minimum measured value.
SP_MAX	Float[8,2]	Maximum setpoint value.
SP_MIN	Float[8,2]	Minimum setpoint value.

Table 3-79. GENBIN object type attributes. (Continued)

Attribute	OTDL Data Type	Description
OP_MAX	Float[8,2]	Maximum output value.
OP_MIN	Float[8,2]	Minimum output value.

Table 3-80. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
IMPLEMENTED	1	Implemented
ALARM_BLK	2	Alarm is blocked.
AL_PE_BLK	3	Alarm period block.
PRINT_BLK	4	Printout is blocked.
REP_FAIL_BLK	5	Repeated fail blocked.
EVENT_BLK	6	Event is blocked.
SELECTED	7	Object is selected.
MORD_EV_BLK	8	Not used.
AU_ST_00	9	Unacknowledged signal error.
AU_ST_01	10	Unacknowledged alarm feedback error.
AU_ST_02	11	Not used.
AU_ST_03	12	Not used.
AU_ST_04	13	Unacknowledged alarm high limit 2.
AU_ST_05	14	Unacknowledged alarm high limit 1.
AU_ST_06	15	Unacknowledged alarm low limit 1.
AU_ST_07	16	Unacknowledged alarm low limit 2.
PC_ST_00	17	Signal error.
PC_ST_01	18	Alarm feedback error.
PC_ST_02	19	Not used.
PC_ST_03	20	Not used.
PC_ST_04	21	Alarm high limit 2.
PC_ST_05	22	Alarm high limit 1.
PC_ST_06	23	Alarm low limit 1.
PC_ST_07	24	Alarm low limit 2.

Table 3-80. TdStatus. Status bits (Continued)

Bits in TdStatus	Bit number	Description
PC_ST_08	25	Interlock mode.
PC_ST_09	26	Interlock process.
PC_ST_10	27	Interlock switch-gear.
PC_ST_11	28	Interlock emergency trip.
PC_ST_12	29	Not used.
PC_ST_13	30	Not used.
PC_ST_14	31	Not used.
PC_ST_15	32	Not used.

Table 3-81. Status2. Status bits

Bits in status2	Bit number	Description
PC_ST_16	17	Central mode.
PC_ST_17	18	Local mode.
PC_ST_18	19	Remote mode.
PC_ST_19	20	Auto mode.
PC_ST_20	21	Manual mode.
PC_ST_21	22	Blocked mode.
PC_ST_22	23	Standby mode.
PC_ST_23	24	Not used.
PC_ST_24	25	Order output ON/Open.
PC_ST_25	26	Order output OFF/Close.
PC_ST_26	27	Not used.
PC_ST_27	28	Not used.
PC_ST_28	29	Not used.
PC_ST_29	30	Not used.
PC_ST_30	31	Not used.

3.10.4 Connections

The GENBIN object does not have any connections because they are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

3.10.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

genbin_ab1_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, L2, NO_OF_DEC

genbin_ab2_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, H1, L1, L2, MV_MAX, MV_MIN, NO_OF_DEC, NAME

genbin_ab3_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, NO_OF_DEC, NAME, DESCRIPTION

genbin_ab4_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, SP_MMC, SP_INTL_MMC, OP_MMC, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, SUBSYSTEM, CLASS, NO_OF_DEC, NAME, DESCRIPTION

3.10.6 Commands

The commands are treated in the same way as for calculated objects in ABB Master Process Stations. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-82. GENBIN operation on the ORDER Command

Type of function for corresponding parameters	Parameters to the ORDER command: Opcode OpProp	Functionality Output Exceptions
Alarm Block Blocks alarms handling for interlocks	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_BLK is set to true.
Alarm Deblock Deblocks alarms handling for interlocks	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked the STATUS_1 bit ALARM_BLK is set to false.
Print Block Blocks printout for events	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS_1 bit PRINT_BLK is set to true.
Print Deblock Deblocks printout for events	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, the STATUS_1 bit PRINT_BLK is set to false.
Event Block Block event	dcBLOCK_DCX dcGENEVENT_DCX	If not already blocked, the STATUS_1 bit EVENT_BLK is set to true.

Table 3-82. GENBIN operation on the ORDER Command (Continued)

Event Deblock Deblocks event	dcDEBLOCK_DCX dcGENEVENT_DCX	If blocked, the STATUS_1 bit EVENT_BLK is set to false.
MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the min and max range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the min and max range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the min and max range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.
MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the min and max range. Must also be smaller than the L1 limit.
On/Open On/open order	dcSET_DCX, dcRESET_DCX dcGENE08BOOL_DCX	MORD_08 = 1 and ORDONOP = 1.
Off/Close Off/close order	dcSET_DCX, dcRESET_DCX dcGENE09BOOL_DCX	MORD_09 = 1 and ORDOFCL = 1.
MAN Set to manual mode	dcSET_DCX, dcRESET_DCX dcGENE04BOOL_DCX	MORD_04 = 1 and ORDMAN = 1.
AUTO Set to auto mode	dcSET_DCX, dcRESET_DCX dcGENE05BOOL_DCX	MORD_05 = 1 and ORDAUTO = 1.
Blocked Control Set to blocked point of control	dcSET_DCX, dcRESET_DCX dcGENE06BOOL_DCX	MORD_06 = 1 and ORDBLK = 1.
Standby Control Set to standby point of control	dcSET_DCX, dcRESET_DCX dcGENE07BOOL_DCX	MORD_07 = 1 and ORDSTBY = 1.
Central Control Set to central point of control	dcSET_DCX, dcRESET_DCX dcGENE01BOOL_DCX	MORD_01 = 1 and ORDCEN = 1.

Table 3-82. GENBIN operation on the ORDER Command (Continued)

Local Control Set to local point of control	dcSET_DCX, dcRESET_DCX dcGENE02BOOL_DCX	MORD_02 = 1 and ORDLOC = 1.
Remote Control Set to remote point of control	dcSET_DCX, dcRESET_DCX dcGENE03BOOL_DCX	MORD_03 = 1 and ORDREM = 1.

3.10.7 Events

The GENBIN object type has one event called EVENT:

Table 3-83. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the GENBIN objects always request for at least the STATUS attribute to be returned.

3.11 General Controller - GENCON

3.11.1 How to Use

The Object Type General Controller, here after called GENCON, exists as process object type in ABB Master Process Station.

The object type GENCON incorporates functions for:

- User defined control function
- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.11.2 Overview

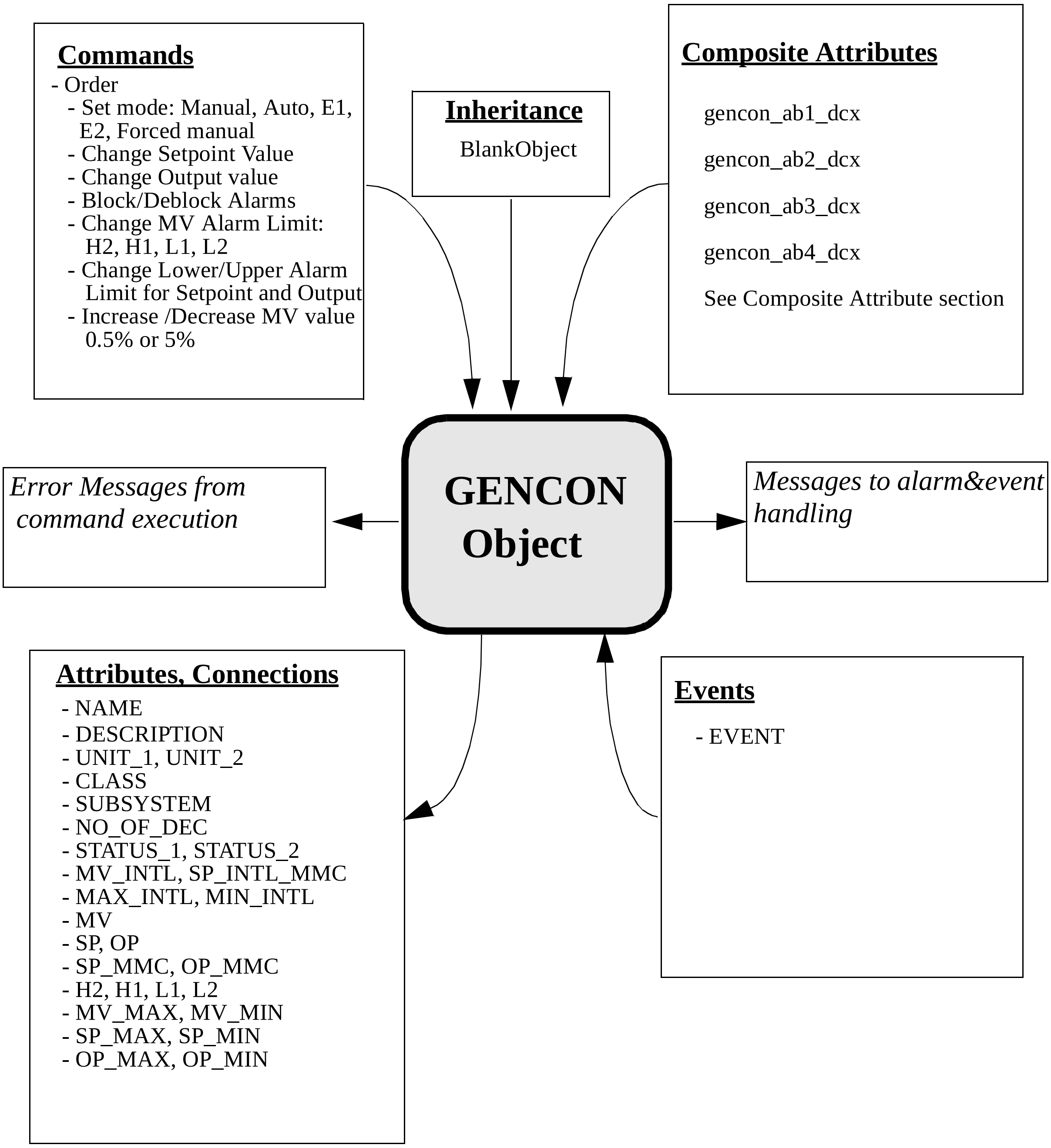


Figure 3-15. Overview of the Object Type GENCON

3.11.3 Attributes

The object attributes are a subset of the corresponding attributes of the GENCONs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master Database Element or Functional Units manuals.

Table 3-84. GENCON object type attributes.

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
UNIT_1	String[7]	The unit of the measured value and setpoint, in percentage, in engineering unit.
UNIT_2	String[7]	The unit of the measured value 2 and setpoint 2, in percentage, in engineering unit.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Process section 0 -16
NO_OF_DEC	IntSmall	Number of decimals.
STATUS_1	TdStatus	Represents the status of the object. See Table 3-85
STATUS_2	Status2	Represents the status of the object. See Table 3-86
MV_INTL	IntLong	Measured integer value.
SP_INTL_MMC	IntLong	Setpoint integer value.
MAX_INTL	IntLong	Maximum integer value.
MIN_INTL	IntLong	Minimum integer value.
MV	Float[8,2]	Measured value.
SP	Float[8,2]	Setpoint value.
OP	Float[8,2]	Output value.
SP_MMC	Float[8,2]	Preferred setpoint value set by the operator.
OP_MMC	Float[8,2]	Preferred output value set by the operator.
H2	Float[8,2]	Upper alarm limit H2.
H1	Float[8,2]	Upper alarm limit H1.
L1	Float[8,2]	Lower alarm limit L1.
L2	Float[8,2]	Lower alarm limit L2.
MV_MAX	Float[8,2]	Maximum measured value.
MV_MIN	Float[8,2]	Minimum measured value.
SP_MAX	Float[8,2]	Maximum setpoint value.
SP_MIN	Float[8,2]	Minimum setpoint value.

Table 3-84. GENCON object type attributes. (Continued)

Attribute	OTDL Data Type	Description
OP_MAX	Float[8,2]	Maximum output value.
OP_MIN	Float[8,2]	Minimum output value.

Table 3-85. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
IMPLEMENTED	1	Implemented
ALARM_BLK	2	Alarm is blocked.
AL_PE_BLK	3	Alarm period block.
PRINT_BLK	4	Printout is blocked.
REP_FAIL_BLK	5	Repeated fail blocked.
EVENT_BLK	6	Event is blocked.
SELECTED	7	Object is selected.
MORD_EV_BLK	8	Not used.
AU_ST_00	9	Unacknowledged signal error.
AU_ST_01	10	Not used.
AU_ST_02	11	Not used.
AU_ST_03	12	Unacknowledged alarm deviation.
AU_ST_04	13	Unacknowledged alarm high limit 2.
AU_ST_05	14	Unacknowledged alarm high limit 1.
AU_ST_06	15	Unacknowledged alarm low limit 1.
AU_ST_07	16	Unacknowledged alarm low limit 2.
PC_ST_00	17	Signal error.
PC_ST_01	18	Not used.
PC_ST_02	19	Not used.
PC_ST_03	20	Alarm deviation.
PC_ST_04	21	Alarm high limit 2.
PC_ST_05	22	Alarm high limit 1.
PC_ST_06	23	Alarm low limit 1.
PC_ST_07	24	Alarm low limit 2.

Table 3-85. TdStatus. Status bits (Continued)

Bits in TdStatus	Bit number	Description
PC_ST_08	25	Gives warning for high setpoint limit.
PC_ST_09	26	Gives warning for low setpoint limit.
PC_ST_10	27	Gives warning for high output limit.
PC_ST_11	28	Gives warning for low output limit.
PC_ST_12	29	Not used.
PC_ST_13	30	Not used.
PC_ST_14	31	Not used.
PC_ST_15	32	Not used.

Table 3-86. Status2. Status bits

Bits in status2	Bit number	Description
PC_ST_16	17	Balanced mode.
PC_ST_17	18	Manual forced mode.
PC_ST_18	19	Manual mode.
PC_ST_19	20	Auto mode.
PC_ST_20	21	External reference E1.
PC_ST_21	22	External reference E2.
PC_ST_22	23	Not used.
PC_ST_23	24	Not used.
PC_ST_24	25	Order output ON/Open.
PC_ST_25	26	Order output OFF/Close.
PC_ST_26	27	Not used.
PC_ST_27	28	Not used.
PC_ST_28	29	Not used.
PC_ST_29	30	Not used.
PC_ST_30	31	Not used.

3.11.4 Connections

The GENCON object does not have any connections because they are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

3.11.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

gencon_ab1_dcx:

> STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, L2, NO_OF_DEC

gencon_ab2_dcx:

> STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, L2, MV_MAX, MV_MIN, NO_OF_DEC, NAME

gencon_ab3_dcx:

> STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, NO_OF_DEC, NAME, DESCRIPTION

gencon_ab4_dcx:

> STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, SP_MMC, SP_INTL_MMC, OP_MMC, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, SUBSYSTEM, CLASS, NO_OF_DEC, NAME, DESCRIPTION

3.11.6 Commands

The commands are treated in the same way as for calculated objects in ABB Master Process Stations. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-87. GENCON operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
SETPOINT Command for changing the setpoint value	dcVALUE_CHANGE_DCX dcGENSPMMC_DCX	Changes the setpoint value.
MAN OUT Command for changing the output value	dcVALUE_CHANGE_DCX dcGENOPMMC_DCX	Changes the output value.
MAN FD Set to forced manual mode	dcSET_DCX, dcRESET_DCX dcGENE02BOOL_DCX	MORD_02 = 1 and ORDMANF = 1.
MAN Set to manual mode	dcSET_DCX, dcRESET_DCX dcGENE03BOOL_DCX	MORD_03 = 1 and ORDMAN = 1.
AUTO Set to auto mode	dcSET_DCX, dcRESET_DCX dcGENE04BOOL_DCX	MORD_04 = 1 and ORDAUTO = 1.
E1 External reference E1	dcSET_DCX, dcRESET_DCX dcGENE05BOOL_DCX	MORD_05 = 1 and ORDE1 = 1.
E2 External reference E2	dcSET_DCX, dcRESET_DCX dcGENE06BOOL_DCX	MORD_06 = 1 and ORDE2 = 1.
Alarm Block Blocks alarms handling for interlocks	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_BLK is set to true.
Alarm Deblock Deblocks alarms handling for interlocks	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked the STATUS_1 bit ALARM_BLK is set to false.
MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the max and min range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the max and min range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the max and min range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.

Table 3-87. GENCON operation on the ORDER Command (Continued)

MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the max and min range. Must also be smaller than the L1 limit.
SETPH Command to change the upper warning limit setpoint	dcVALUE_CHANGE_DCX dcGENSPMAX_DCX	Changes the upper warning limit setpoint.
SETPL Command to change the lower warning limit setpoint	dcVALUE_CHANGE_DCX dcGENSPMIN_DCX	Changes the lower warning limit setpoint.
OUTPH Command to change the upper warning limit output	dcVALUE_CHANGE_DCX dcGENOPMAX_DCX	Changes the upper warning limit output.
OUTPL Command to change the lower warning limit output	dcVALUE_CHANGE_DCX dcGENOPMIN_DCX	Changes the lower warning limit output.
Small Increase Increase the output value with 0.5% of range	dcSMALL_INCREASE_DCX dcACT_VALUE_DCX	Value is increased with 0.5% of total range
Large Increase Increase the output value with 5% of range	dcLARGE_INCREASE_DCX dcACT_VALUE_DCX	Value is increased with 5% of total range.
Small Decrease Decrease the output value with 0.5% of range	dcSMALL_DECREASE_DCX dcACT_VALUE_DCX	Value is decreased with 0.5% of total range.
Large Decrease Decrease the output value with 5% of range	dcLARGE_DECREASE_DCX dcACT_VALUE_DCX	Value is decreased with 5% of total range.

3.11.7 Events

The GENCON object type has one event called EVENT:

Table 3-88. The event EVENT

Name	**Value/explanation**
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the GENCON objects always request for at least the STATUS attribute to be returned.

3.12 General User Defined Object - GENUSD

3.12.1 How to Use

The Object Type General User Defined Object, here after called GENUSD, exists as process object type in ABB Master Process Station.

The object type GENUSD incorporates functions for:

- User defined control function
- Limit checking
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.12.2 Overview

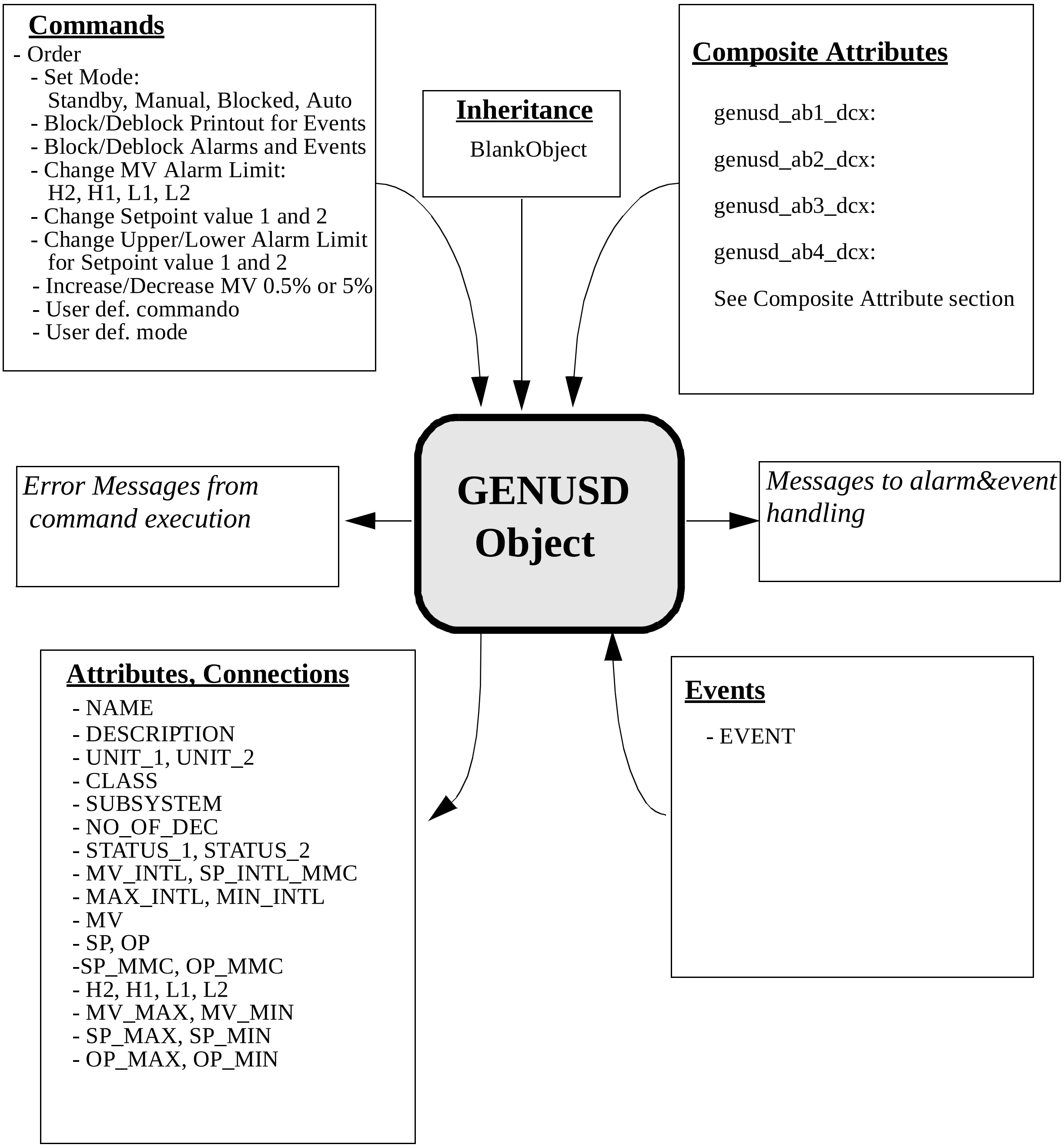


Figure 3-16. Overview of the Object Type GENUSD

3.12.3 Attributes

The object attributes are a subset of the corresponding attributes of the GENUSDs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master Database Element or Functional Units manuals

Table 3-89. GENUSD object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
UNIT_1	String[7]	The unit of the measured value and setpoint, in percentage, in engineering unit.
UNIT_2	String[7]	The unit of the measured value 2 and setpoint 2, in percentage, in engineering unit.
CLASS	IntSmall	Object class.
SUBSYSTEM	IntSmall	Process section 0 -16.
NO_OF_DEC	IntSmall	Number of decimals.
STATUS_1	TdStatus	Represents the status of the object. See Table 3-90
STATUS_2	Status2	Represents the status of the object. See Table 3-91
MV_INTL	IntLong	Measured integer value.
SP_INTL_MMC	IntLong	Setpoint integer value.
MAX_INTL	IntLong	Maximum integer value.
MIN_INTL	IntLong	Minimum integer value.
MV	Float[8,2]	Measured value.
SP	Float[8,2]	Setpoint value.
OP	Float[8,2]	Output value.
SP_MMC	Float[8,2]	Preferred setpoint value set by the operator.
OP_MMC	Float[8,2]	Preferred output value set by the operator.
H2	Float[8,2]	Upper alarm limit H2.
H1	Float[8,2]	Upper alarm limit H1.
L1	Float[8,2]	Lower alarm limit L1.
L2	Float[8,2]	Lower alarm limit L2.
MV_MAX	Float[8,2]	Maximum measured value.
MV_MIN	Float[8,2]	Minimum measured value.
SP_MAX	Float[8,2]	Maximum setpoint value.
SP_MIN	Float[8,2]	Minimum setpoint value.

Table 3-89. GENUSD object type attributes (Continued)

Attribute	OTDL Data Type	Description
OP_MAX	Float[8,2]	Maximum output value.
OP_MIN	Float[8,2]	Minimum output value.

Table 3-90. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
IMPLEMENTED	1	Implemented
ALARM_BLK	2	Alarm is blocked.
AL_PE_BLK	3	Alarm period block.
PRINT_BLK	4	Printout is blocked.
REP_FAIL_BLK	5	Repeats failed block
EVENT_BLK	6	Event is blocked.
SELECTED	7	Object is selected.
MORD_EV_BLK	8	Not used.
AU_ST_00	9	Unacknowledged signal error 1.
AU_ST_01	10	Unacknowledged signal error 2.
AU_ST_02	11	Unacknowledged alarm 1.
AU_ST_03	12	Unacknowledged alarm 2.
AU_ST_04	13	Unacknowledged alarm 3.
AU_ST_05	14	Unacknowledged alarm 4.
AU_ST_06	15	Unacknowledged alarm 5.
AU_ST_07	16	Unacknowledged alarm 6.
PC_ST_00	17	Signal error 1.
PC_ST_01	18	Signal error 2.
PC_ST_02	19	Alarm 1.
PC_ST_03	20	Alarm 2.
PC_ST_04	21	Alarm 3.
PC_ST_05	22	Alarm 4.
PC_ST_06	23	Alarm 5.
PC_ST_07	24	Alarm 6.
PC_ST_08	25	Interlock 1.

Table 3-90. TdStatus. Status bits (Continued)

Bits in TdStatus	Bit number	Description
PC_ST_09	26	Interlock 2.
PC_ST_10	27	Interlock 3.
PC_ST_11	28	Interlock 4.
PC_ST_12	29	Interlock 5.
PC_ST_13	30	Interlock 6.
PC_ST_14	31	Not used.
PC_ST_15	32	Not used.

Table 3-91. Status2. Status bits

Bits in status2	Bit number	Description
PC_ST_16	17	Operator position M1.
PC_ST_17	18	Operator position M2.
PC_ST_18	19	Operator position M3.
PC_ST_19	20	Auto mode.
PC_ST_20	21	Manual mode.
PC_ST_21	22	Blocked mode.
PC_ST_22	23	Standby mode.
PC_ST_23	24	Not used.
PC_ST_24	25	On/Off square indication.
PC_ST_25	26	On/Off square indication.
PC_ST_26	27	On/Off square indication.
PC_ST_27	28	On/Off square indication.
PC_ST_28	29	Not used.
PC_ST_29	30	Not used.
PC_ST_30	31	Not used.

3.12.4 Connections

The GENUSD object does not have any connections because they are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

3.12.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

genusd_ab1_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, L2, NO_OF_DEC

genusd_ab2_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, L2, MV_MAX, MV_MIN, NO_OF_DEC, NAME

genusd_ab3_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, NO_OF_DEC, NAME, DESCRIPTION

genusd_ab4_dcx:

STATUS_1, STATUS_2, MV, MV_INTL, SP, OP, SP_MMC, SP_INTL_MMC, OP_MMC, H2, H1, L1, L2, MV_MAX, MV_MIN, SP_MAX, SP_MIN, OP_MAX, OP_MIN, MAX_INTL, MIN_INTL, UNIT_1, UNIT_2, SUBSYSTEM, CLASS, NO_OF_DEC, NAME, DESCRIPTION

3.12.6 Commands

The commands are treated in the same way as for calculated objects in ABB Master Process Stations. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-92. GENUSD operation on the ORDER Command.

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
C1 User def command 1	dcSET_DCX, dcRESET_DCX dcGENE09BOOL_DCX	MORD_09 = 1 and ORDC1 = 1.
C2 User def command 2	dcSET_DCX, dcRESET_DCX dcGENE10BOOL_DCX	MORD_10 = 1 and ORDC2 = 1.
C3 User def command 3	dcSET_DCX, dcRESET_DCX dcGENE11BOOL_DCX	MORD_11 = 1 and ORDC3 = 1.
C4 User def command 4	dcSET_DCX, dcRESET_DCX dcGENE12BOOL_DCX	MORD_12 = 1 and ORDC4 = 1.
C5 User def command 5	dcSET_DCX, dcRESET_DCX dcGENE13BOOL_DCX	MORD_13 = 1 and ORDC5 = 1.
C6 User def command 6	dcSET_DCX, dcRESET_DCX dcGENE14BOOL_DCX	MORD_14 = 1 and ORDC6 = 1.

Table 3-92. GENUSD operation on the ORDER Command. (Continued)

C7 User def command 7	dcSET_DCX, dcRESET_DCX dcGENE15BOOL_DCX	MORD_15 = 1 and ORDC7 = 1.
C8 User def command 8	dcSET_DCX, dcRESET_DCX dcGENE16BOOL_DCX	MORD_16 = 1 and ORDC8 = 1.
SETPOINT Command for changing the setpoint value 1	dcVALUE_CHANGE_DCX dcGENSPMMC_DCX	Changes setpoint value 1.
SETPOINT 2 Command for changing the setpoint value 2	dcVALUE_CHANGE_DCX dcGENSPIMMC_DCX	Changes setpoint value 2.
MAN Set to manual mode	dcSET_DCX, dcRESET_DCX dcGENE04BOOL_DCX	MORD_04 = 1 and ORDMAN = 1.
AUTO Set to auto mode	dcSET_DCX, dcRESET_DCX dcGENE05BOOL_DCX	MORD_05 = 1 and ORDAUTO = 1.
Blocked Control Set to blocked point of control	dcSET_DCX, dcRESET_DCX dcGENE06BOOL_DCX	MORD_06 = 1 and ORDBLK = 1.
Standby Control Set to standby point of control	dcSET_DCX, dcRESET_DCX dcGENE07BOOL_DCX	MORD_07 = 1 and ORDSTBY = 1.
M1 User def mode 1	dcSET_DCX, dcRESET_DCX dcGENE01BOOL_DCX	MORD_01 = 1 and ORDM1 = 1.
M2 User def mode 2	dcSET_DCX, dcRESET_DCX dcGENE02BOOL_DCX	MORD_02 = 1 and ORDM2 = 1.
M3 User def mode 3	dcSET_DCX, dcRESET_DCX dcGENE03BOOL_DCX	MORD_03 = 1 and ORDM3 = 1.
Alarm Block Blocks alarms handling for interlocks	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS_1 bit ALARM_BLK is set to true.
Alarm Deblock Deblocks alarms handling for interlocks	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked the STATUS_1 bit ALARM_BLK is set to false.
Print Block Blocks printout for events	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS_1 bit PRINT_BLK is set to true.
Print Deblock Deblocks printout for events	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, the STATUS_1 bit PRINT_BLK is set to false.

Table 3-92. GENUSD operation on the ORDER Command. (Continued)

Event Block Block event	dcBLOCK_DCX dcGENEVENT_DCX	If not already blocked, the STATUS_1 bit EVENT_BLK is set to true.
Event Deblock Deblock event	dcDEBLOCK_DCX dcGENEVENT_DCX	If blocked, the STATUS_1 bit EVENT_BLK is set to false.
MVH2 Command to change the measure value alarm limit H2	dcVALUE_CHANGE_DCX dcHILIM2_DCX	Changes the measure value alarm limit H2. Must be within the max and min range. Must also be larger than any other valid limit.
MVH1 Command to change the measure value alarm limit H1	dcVALUE_CHANGE_DCX dcHILIM1_DCX	Changes the measure value alarm limit H1. Must be within the max and min range. Must also be smaller than the H2 limit and larger than the L1 and L2 limits.
MVL1 Command to change the measure value alarm limit L1	dcVALUE_CHANGE_DCX dcLOLIM1_DCX	Changes the measure value alarm limit L1. Must be within the max and min range. Must also be smaller than the H1 and H2 limits, but larger than the L2 limit.
MVL2 Command to change the measure value alarm limit L2	dcVALUE_CHANGE_DCX dcLOLIM2_DCX	Changes the measure value alarm limit L2. Must be within the max and min range. Must also be smaller than the L1 limit.
SETPH Command to change the upper warning limit setpoint	dcVALUE_CHANGE_DCX dcGENSPMAX_DCX	Changes the upper warning limit setpoint.
SETPL Command to change the lower warning limit setpoint	dcVALUE_CHANGE_DCX dcGENSPMIN_DCX	Changes the lower warning limit setpoint.
SETP2H Command to change the upper warning limit setpoint value 2	dcVALUE_CHANGE_DCX dcGENMAXINTL_DCX	Changes the upper warning limit setpoint value 2.
SETP2L Command to change the lower warning limit setpoint value	dcVALUE_CHANGE_DCX dcGENMININTL_DCX	Changes the lower warning limit setpoint value 2.
Small Increase Increase the output value with 0.5% of range	dcSMALL_INCREASE_DCX dcACT_VALUE_DCX	Value is increased with 0.5% of total range.

Table 3-92. GENUSD operation on the ORDER Command. (Continued)

Large Increase Increase the output value with 5% of range	dcLARGE_INCREASE_DCX dcACT_VALUE_DCX	Value is increased with 5% of total range.
Small Decrease Decrease the output value with 0.5% of range	dcSMALL_DECREASE_DCX dcACT_VALUE_DCX	Value is decreased with 0.5% of total range.
Large Decrease Decrease the output value with 5% of range	dcLARGE_DECREASE_DCX dcACT_VALUE_DCX	Value is decreased with 5% of total range.

3.12.7 Events

The GENUSD object type has one event called EVENT:

Table 3-93. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the GENUSD objects always request for at least the STATUS attribute to be returned.

3.13 Valve Control - VALVECON

3.13.1 How to Use

The Object Type Valve Control, here after called VALVECON, exists as process object type in ABB Master Process Station.

The object type VALVECON incorporates functions for:

- Evaluation of interlockings
- Open/close control
- Operation time supervision
- Control from a local operator’s console
- MAN/AUTO operation
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.13.2 Overview

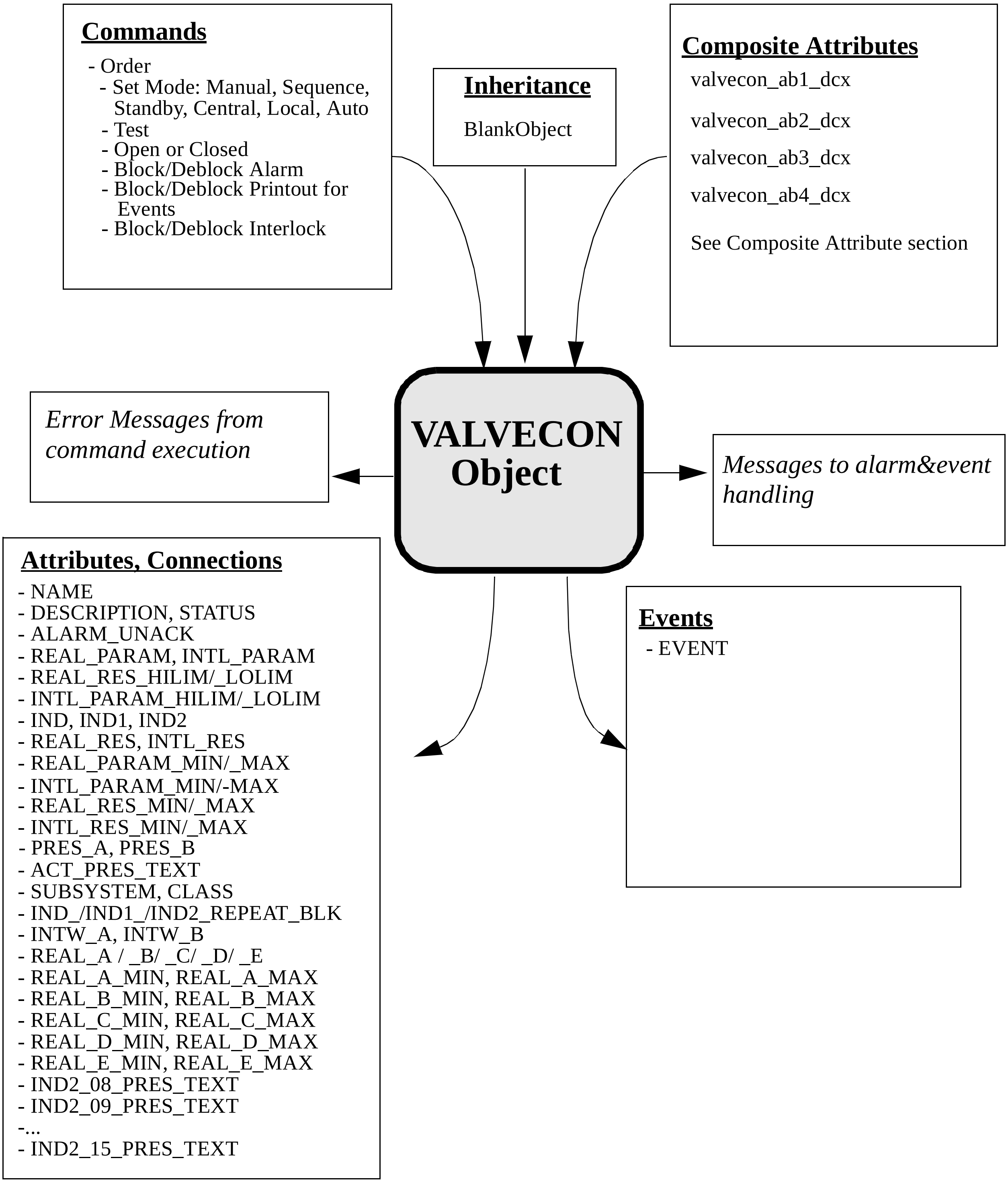


Figure 3-17. Overview of the Object Type VALVECON

3.13.3 Attributes

The object attributes are a subset of the corresponding attributes of the VALVECONs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master data base element or Functional Unit manuals

Table 3-94. VALVECON object type attributes.

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the object. See Table 3-95
ALARM_UNACK	alarm_unack	Alarm unacknowledged. See Table 3-99
REAL_PARAM	Float[8,2]	A real value, used by AMPL.
INTL_PARAM	IntLong	An integer value, used by AMPL.
REAL_RES_HILIM	Float[8,2]	A high limit value used by AMPL to make limit detection on a real value, for instance REAL_RES. Nominal current of motor.
REAL_RES_LOLIM	Float[8,2]	A low limit value used by AMPL to make limit detection on a real value, for instance REAL_RES.
IND	ind	See Table 3-100
IND1	IntShort	Indication 1 terminal.
IND2	IntShort	Indication 2 terminal.
REAL_RES	Float[8,2]	Motor current in percentage of rated current. A real value from AMPL.
INTL_RES	IntLong	An integer value from AMPL.
REAL_PARAM_MIN	Float[8,2]	Minimum value for REAL_PARAM.
REAL_PARAM_MAX	Float[8,2]	Maximum value for REAL_PARAM.
INTL_PARAM_MIN	IntLong	Minimum value for INTL_PARAM.
INTL_PARAM_MAX	IntLong	Maximum value for INTL_PARAM.
REAL_RES_MIN	Float[8,2]	Shows the lowest of the motor’s measure values in percentage of the motor’s rated current.
REAL_RES_MAX	Float[8,2]	Shows the highest of the motor’s measure values in percentage of the motor’s rated current.
INTL_RES_MIN	IntLong	Minimum value for INTL_RES.
INTL_RES_MAX	IntLong	Maximum value for INTL_RES.
PRES_A	String[11]	Presentation of position A.
PRES_B	String[11]	Presentation of position B.

Table 3-94. VALVECON object type attributes. (Continued)

Attribute	OTDL Data Type	Description
ACT_PRES_TEXT	String[21]	Presentation of active interlocks. The string comes from one of the texts IND2_08_PRES_TEXT to IND2_14_PRES_TEXT. The text is prioritized by the active signal with the lowest number among the signals IND2_08 toIND2_14.
SUBSYSTEM	IntSmall	Process section 0 -16.
CLASS	IntSmall	Object class.
IND_REPEAT_BLK	ind_repeat_blk	
IND1_REPEAT_BLK	IntShort	Shows which of the signals IND1_00 to IND1_15 that have repeated error blocks. See Table 3-96
IND2_REPEAT_BLK	IntShort	Shows which of the signals IND2_00 to IND2_15 that have repeated error blocks. See Table 3-96
INTW_A	intw_a	Additional integers. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically to the presentation interpreter. See Table 3-97
INTW_B	intw_b	Additional integers. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically to the presentation interpreter. See Table 3-98
REAL_A	Float[8,2]	Additional reals. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_B	Float[8,2]	Additional reals. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_C	Float[8,2]	Additional reals. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_D	Float[8,2]	Additional reals. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_E	Float[8,2]	Additional reals. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_A_MIN	Float[8,2]	Minimum value for REAL_A.
REAL_A_MAX	Float[8,2]	Maximum value for REAL_A.
REAL_B_MIN	Float[8,2]	Minimum value for REAL_B.
REAL_B_MAX	Float[8,2]	Maximum value for REAL_B.
REAL_C_MIN	Float[8,2]	Minimum value for REAL_C.
REAL_C_MAX	Float[8,2]	Maximum value for REAL_C.
REAL_D_MIN	Float[8,2]	Minimum value for REAL_D.
REAL_D_MAX	Float[8,2]	Maximum value for REAL_D.
REAL_E_MIN	Float[8,2]	Minimum value for REAL_E.

Table 3-94. VALVECON object type attributes. (Continued)

Attribute	OTDL Data Type	Description
REAL_E_MAX	Float[8,2]	Maximum value for REAL_E.
IND2_08_PRES_TEXT	String[21]	For presentation of C-interlock 1 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_08. Highest priority.
IND2_09_PRES_TEXT	String[21]	For presentation of C-interlock 2 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_09.
IND2_10_PRES_TEXT	String[21]	For presentation of B-interlock 3 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_10.
IND2_11_PRES_TEXT	String[21]	For presentation of B-interlock 4 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_11.
IND2_12_PRES_TEXT	String[21]	For presentation of B-interlock 1 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_12.
IND2_13_PRES_TEXT	String[21]	For presentation of B-interlock 2 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_13.
IND2_14_PRES_TEXT	String[21]	For presentation of A-interlock in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_14. Lowest priority.
IND2_15_PRES_TEXT	String[21]	Not used.

Table 3-95. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
IMPLEMENTED	1	Implemented
ALARM_BLK	2	Alarm is blocked.
ALARM_PERIOD_BLK	3	Alarm period block.
PRINT_BLK	4	Printout is blocked.
IND1_DIST	5	IND1 disturbance. Controls the color priority at fails. True when at least one property within the block property IND1 is true.
IND2_DIST	6	IND2 disturbance. Controls the color priority at fails. True when at least one property within the block property IND2 is true.
SELECTED	7	Object selected by the operator.
AU_IND	8	Indicates whether there exists unacknowledged alarms. True when at least one bit in the property ALARM_UNACK is set.
REPEAT_BLK	9	Blocks repeated fails active/not active.
OH_SPARE_01	10	Spare property for future object handling functions. Will be sent together with the STATUS property.

Table 3-95. TdStatus. Status bits (Continued)

Bits in TdStatus	**Bit number**	**Description**
OH_SPARE_02	11	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_03	12	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_04	13	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_05	14	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_06	15	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_07	16	Spare property for future object handling functions. Will be sent together with the STATUS property.
BOOL_A	17	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_B	18	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_C	19	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_D	20	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_E	21	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_F	22	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_G	23	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_H	24	Additional Booleans. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.

Table 3-96. Ind_repeat_blk. Status bits

Bits in ind_repeat_blk	Bit number	Description
IND1_00_REBLK	1	Blocks repeated alarms for bit number 1 in the IND1 attribute.
IND1_01_REBLK	2	Blocks repeated alarms for bit number 2in the IND1 attribute.
IND1_02_REBLK	3	Blocks repeated alarms for bit number 3 in the IND1 attribute.
IND1_03_REBLK	4	Blocks repeated alarms for bit number 4 in the IND1 attribute.
IND1_04_REBLK	5	Blocks repeated alarms for bit number 5 in the IND1 attribute.
IND1_05_REBLK	6	Blocks repeated alarms for bit number 6 in the IND1 attribute.
IND1_06_REBLK	7	Blocks repeated alarms for bit number 7 in the IND1 attribute.
IND1_07_REBLK	8	Blocks repeated alarms for bit number 8 in the IND1 attribute.
IND1_08_REBLK	9	Blocks repeated alarms for bit number 9 in the IND1 attribute.
IND1_09_REBLK	10	Blocks repeated alarms for bit number 10 in the IND1 attribute.
IND1_10_REBLK	11	Blocks repeated alarms for bit number 11 in the IND1 attribute.
IND1_11_REBLK	12	Blocks repeated alarms for bit number 12 in the IND1 attribute.
IND1_12_REBLK	13	Blocks repeated alarms for bit number 13 in the IND1 attribute.
IND1_13_REBLK	14	Blocks repeated alarms for bit number 14 in the IND1 attribute.
IND1_14_REBLK	15	Blocks repeated alarms for bit number 15 in the IND1 attribute.
IND1_15_REBLK	16	Blocks repeated alarms for bit number 16 in the IND1 attribute.
IND2_00_REBLK	17	Blocks repeated alarms for bit number 1 in the IND2 attribute.
IND2_01_REBLK	18	Blocks repeated alarms for bit number 2 in the IND2 attribute.
IND2_02_REBLK	19	Blocks repeated alarms for bit number 3 in the IND2 attribute.
IND2_03_REBLK	20	Blocks repeated alarms for bit number 4 in the IND2 attribute.
IND2_04_REBLK	21	Blocks repeated alarms for bit number 5 in the IND2 attribute.
IND2_05_REBLK	22	Blocks repeated alarms for bit number 6 in the IND2 attribute.
IND2_06_REBLK	23	Blocks repeated alarms for bit number 7 in the IND2 attribute.
IND2_07_REBLK	24	Blocks repeated alarms for bit number 8 in the IND2 attribute.
IND2_08_REBLK	25	Blocks repeated alarms for bit number 9 in the IND2 attribute.
IND2_09_REBLK	26	Blocks repeated alarms for bit number 10 in the IND2 attribute.
IND2_10_REBLK	27	Blocks repeated alarms for bit number 11 in the IND2 attribute.
IND2_11_REBLK	28	Blocks repeated alarms for bit number 12 in the IND2 attribute.
IND2_12_REBLK	29	Blocks repeated alarms for bit number 13 in the IND2 attribute.
IND2_13_REBLK	30	Blocks repeated alarms for bit number 14 in the IND2 attribute.

Table 3-96. Ind_repeat_blk. Status bits (Continued)

Bits in ind_repeat_blk	Bit number	Description
IND2_14_REBLK	31	Blocks repeated alarms for bit number 15 in the IND2 attribute.
IND2_15_REBLK	32	Blocks repeated alarms for bit number 16 in the IND2 attribute.

The status bits in the data types intw_a and intw_b can be defined by the user. They are not defined in the standard function.

Table 3-97. Intw_a. Status bits

Bits in intw_a	Bit number	Description
INTW_A_00	1	
INTW_A_01	2	
INTW_A_02	3	
INTW_A_03	4	
INTW_A_04	5	
INTW_A_05	6	
INTW_A_06	7	
INTW_A_07	8	
INTW_A_08	9	
INTW_A_09	10	
INTW_A_10	11	
INTW_A_11	12	
INTW_A_12	13	
INTW_A_13	14	
INTW_A_14	15	
INTW_A_15	16	

Table 3-98. Intw_b. Status bits

Bits in intw_b	Bit number	Description
INTW_B_00	1	
INTW_B_01	2	
INTW_B_02	3	
INTW_B_03	4	

Table 3-98. Intw_b. Status bits

Bits in intw_b	**Bit number**	**Description**
INTW_B_04	5	
INTW_B_05	6	
INTW_B_06	7	
INTW_B_07	8	
INTW_B_08	9	
INTW_B_09	10	
INTW_B_10	11	
INTW_B_11	12	
INTW_B_12	13	
INTW_B_13	14	
INTW_B_14	15	
INTW_B_15	16	

Table 3-99. Alarm_unack. Status bits

Bits in alarm_unack	Bit number	Description
AU_IND1_00	1	Indicates that there is an unacknowledged alarm for bit number 1 in the IND1 attribute.
AU_IND1_01	2	Indicates that there is an unacknowledged alarm for bit number 2 in the IND1 attribute.
AU_IND1_02	3	Indicates that there is an unacknowledged alarm for bit number 3 in the IND1 attribute.
AU_IND1_03	4	Indicates that there is an unacknowledged alarm for bit number 4 in the IND1 attribute.
AU_IND1_04	5	Indicates that there is an unacknowledged alarm for bit number 5 in the IND1 attribute.
AU_IND1_05	6	Indicates that there is an unacknowledged alarm for bit number 6 in the IND1 attribute.
AU_IND1_06	7	Indicates that there is an unacknowledged alarm for bit number 7 in the IND1 attribute.
AU_IND1_07	8	Indicates that there is an unacknowledged alarm for bit number 8 in the IND1 attribute.
AU_IND1_08	9	Indicates that there is an unacknowledged alarm for bit number 9 in the IND1 attribute.
AU_IND1_09	10	Indicates that there is an unacknowledged alarm for bit number 10 in the IND1 attribute.
AU_IND1_10	11	Indicates that there is an unacknowledged alarm for bit number 11 in the IND1 attribute.
AU_IND1_11	12	Indicates that there is an unacknowledged alarm for bit number 12 in the IND1 attribute.
AU_IND1_12	13	Indicates that there is an unacknowledged alarm for bit number 13 in the IND1 attribute.
AU_IND1_13	14	Indicates that there is an unacknowledged alarm for bit number 14 in the IND1 attribute.
AU_IND1_14	15	Indicates that there is an unacknowledged alarm for bit number 15 in the IND1 attribute.
AU_IND1_15	16	Indicates that there is an unacknowledged alarm for bit number 16 in the IND1 attribute.
AU_IND1_16	17	Indicates that there is an unacknowledged alarm for bit number 17 in the IND1attribute.
AU_IND1_17	18	Indicates that there is an unacknowledged alarm for bit number 18 in the IND1 attribute.

Table 3-99. Alarm_unack. Status bits (Continued)

Bits in alarm_unack	Bit number	Description
AU_IND1_18	19	Indicates that there is an unacknowledged alarm for bit number 19 in the IND1 attribute.
AU_IND1_19	20	Indicates that there is an unacknowledged alarm for bit number 20 in the IND1 attribute.
AU_IND1_20	21	Indicates that there is an unacknowledged alarm for bit number 21 in the IND1 attribute.
AU_IND1_21	22	Indicates that there is an unacknowledged alarm for bit number 22 in the IND1 attribute.
AU_IND1_22	23	Indicates that there is an unacknowledged alarm for bit number 23 in the IND1 attribute.
AU_IND1_23	24	Indicates that there is an unacknowledged alarm for bit number 24 in the IND1 attribute.
AU_IND1_24	25	Indicates that there is an unacknowledged alarm for bit number 25 in the IND1 attribute.
AU_IND1_25	26	Indicates that there is an unacknowledged alarm for bit number 26 in the IND1 attribute.
AU_IND1_26	27	Indicates that there is an unacknowledged alarm for bit number 27 in the IND1 attribute.
AU_IND1_27	28	Indicates that there is an unacknowledged alarm for bit number 28 in the IND1 attribute.
AU_IND1_28	29	Indicates that there is an unacknowledged alarm for bit number 29 in the IND1 attribute.
AU_IND1_29	30	Indicates that there is an unacknowledged alarm for bit number 30 in the IND1 attribute.
AU_IND1_30	31	Indicates that there is an unacknowledged alarm for bit number 31 in the IND1 attribute.
AU_IND1_31	32	Indicates that there is an unacknowledged alarm for bit number 32 in the IND1 attribute.

Table 3-100. Ind. Status bits

Bits in ind	Bit number	Description
IND1_00	1	User defined fault 1 coming/passing.
IND1_01	2	User defined fault 2 coming/passing.
IND1_02	3	Position error open coming/passing.
IND1_03	4	Position error closed coming/passing.
IND1_04	5	Limit passed.
IND1_05	6	
IND1_06	7	Limit switch for open position.
IND1_07	8	Collective fault indication.
IND1_08	9	Collective unacknowledged fault indication.
IND1_09	10	
IND1_10	11	
IND1_11	12	Limit switch for open position.
IND1_12	13	Open.
IND1_13	14	Close.
IND1_14	15	
IND1_15	16	Intermediate position.
IND2_00	17	Blocked IB1 and IB3.
IND2_01	18	Control mode LOCAL on/off.
IND2_02	19	Control mode TEST on/off.
IND2_03	20	Control mode STANDBY on/off.
IND2_04	21	Control mode SEQUENCE on/off.
IND2_05	22	Operation mode AUTO on/off.
IND2_06	23	
IND2_07	24	Ready to run.
IND2_08	25	Interlock IC1 on/off.
IND2_09	26	Interlock IC2 on/off.
IND2_10	27	Interlock IB1 on/off.
IND2_11	28	Interlock IB2 on/off.
IND2_12	29	Interlock IB3 on/off.
IND2_13	30	Interlock IB4 on/off.

Table 3-100. Ind. Status bits (Continued)

Bits in ind	Bit number	Description
IND2_14	31	Interlock IA on/off.
IND2_15	32	

3.13.4 Connections

The VALVECON object does not have any connections because they are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

3.13.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

valvecon_ab1_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, ACT_PRES_TEXT, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E

valvecon_ab2_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E

valvecon_ab3_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_PARAM, INTL_PARAM, REAL_RES_HILIM, REAL_RES_LOLIM, REAL_RES_MIN, REAL_RES_MAX, INTL_RES_MIN, INTL_RES_MAX, REAL_A_MIN, REAL_A_MAX, REAL_B_MIN, REAL_B_MAX, REAL_C_MIN, REAL_C_MAX, REAL_D_MIN, REAL_D_MAX, REAL_E_MIN, REAL_E_MAX, IND1_REPEAT_BLK, IND2_REPEAT_BLK, NAME, DESCRIPTION

valvecon_ab4_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_PARAM, INTL_PARAM, REAL_RES_HILIM, REAL_RES_LOLIM, REAL_RES_MIN, REAL_RES_MAX, INTL_RES_MIN, INTL_RES_MAX, REAL_A_MIN, REAL_A_MAX, REAL_B_MIN, REAL_B_MAX, REAL_C_MIN, REAL_C_MAX, REAL_D_MIN, REAL_D_MAX, REAL_E_MIN, REAL_E_MAX, IND1_REPEAT_BLK, IND2_REPEAT_BLK, NAME, DESCRIPTION, IND2_08_PRES_TEXT, IND2_09_PRES_TEXT, IND2_10_PRES_TEXT, IND2_11_PRES_TEXT, IND2_12_PRES_TEXT, IND2_13_PRES_TEXT, IND2_14_PRES_TEXT

3.13.6 Commands

The commands are treated in the same way as for calculated objects in ABB Master Process Stations. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-101. VALVECON operation on the ORDER Command

Type of function for corresponding parameters	Parameters to the ORDER command: Opcode OpProp	Functionality Output Exceptions
MAN Set to manual mode	dcSET_DCX, dcRESET_DCX dcGENE06BOOL_DCX	The valve may be controlled directly from the selected point of control, provided the point of control is not STANDBY.
AUTO Set to auto mode	dcSET_DCX, dcRESET_DCX dcGENE05BOOL_DCX	The process controls the valve by way of an AMPL program. This mode is e. g. used for level control using level transmitters. The part for AUTO is included by setting the call parameter C2 of the AMPL element to ‘1’.
OPEN Order open	dcSET_DCX, dcRESET_DCX dcGENE09BOOL_DCX	MORD_08 = 1.
CLOSE Order close	dcSET_DCX, dcRESET_DCX dcGENE10BOOL_DCX	MORD_09 = 1.
Central Control Set to central point of control	dcSET_DCX, dcRESET_DCX dcGENE04BOOL_DCX	The valve is controlled from MasterView 800, AS 500 or Advant Controller 4xx. Control from other locations are blocked. MORD_03 = 1.
Sequence Control Set to sequence point of control	dcSET_DCX, dcRESET_DCX dcGENE11BOOL_DCX	SEQUENCE is the default point of control. The valve is controlled from a superior AMPL program, e.g. for group start or level control. Control from other locations are blocked. MORD_10 = 1.
Local Control Set to local point of control	dcSET_DCX, dcRESET_DCX dcGENE01BOOL_DCX	The valve is controlled from a local desk or panel. Control from other locations are blocked. The choice of local control is enabled by setting the call parameter C1 of the AMPL element to ‘1’. MORD_00 = 1.
Test Set to test point of control	dcSET_DCX, dcRESET_DCX dcGENE02BOOL_DCX	The valve is controlled from an operator station adjacent to the valve. Is suitable for testing the valve in the field. The control modes MAN and AUTO are not available in point of control TEST. Control from other locations are blocked. MORD_01 = 1.

Table 3-101. VALVECON operation on the ORDER Command (Continued)

Out of Service Set to standby point of control	dcSET_DCX, dcRESET_DCX dcGENE03BOOL_DCX	The valve is brought to its home position and all orders to it are blocked. MORD_02 = 1.
Alarm Block Blocks alarms handling for interlocks	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS bit ALARM_BLK is set to true.
Alarm Deblock Deblocks alarms handling for interlocks	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked the STATUS bit ALARM_BLK is set to false.
Print Block Blocks printout for events	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS bit PRINT_BLK is set to true.
Print Deblock Deblocks printout for events	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked, the STATUS bit PRINT_BLK is set to false.
Interlock Block	dcBLOCK_DCX dcGENE07BOOL_DCX	All A and B interlockings are blocked when the TEST point of control is selected.
Interlock Activated	dcDEBLOCK_DCX dcGENE08BOOL_DCX	An interlocking is active when the AMPL-input is 0 (false).

3.13.7 Events

The VALVECON object type has one event called EVENT.

Table 3-102. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the VALVECON objects always request for at least the STATUS attribute to be returned.

3.14 Motor Control - MOTCON

3.14.1 How to Use

The Object Type Motor Control, here after called MOTCON, exists as process object type in ABB Master Process Station.

The object type MOTCON incorporates functions for:

- Supervision of control circuits
- Evaluation of interlockings

- Control of on/off
- Supervision of motor current
- Forward/reverse running or selection of high/low speed
- Control and supervision of external functions
- Support for subscription requests (on demand, on event, 3 cyclic intervals)
- Alarm detection and handling
- Services to treat Commands
- Update blocking and input

3.14.2 Overview

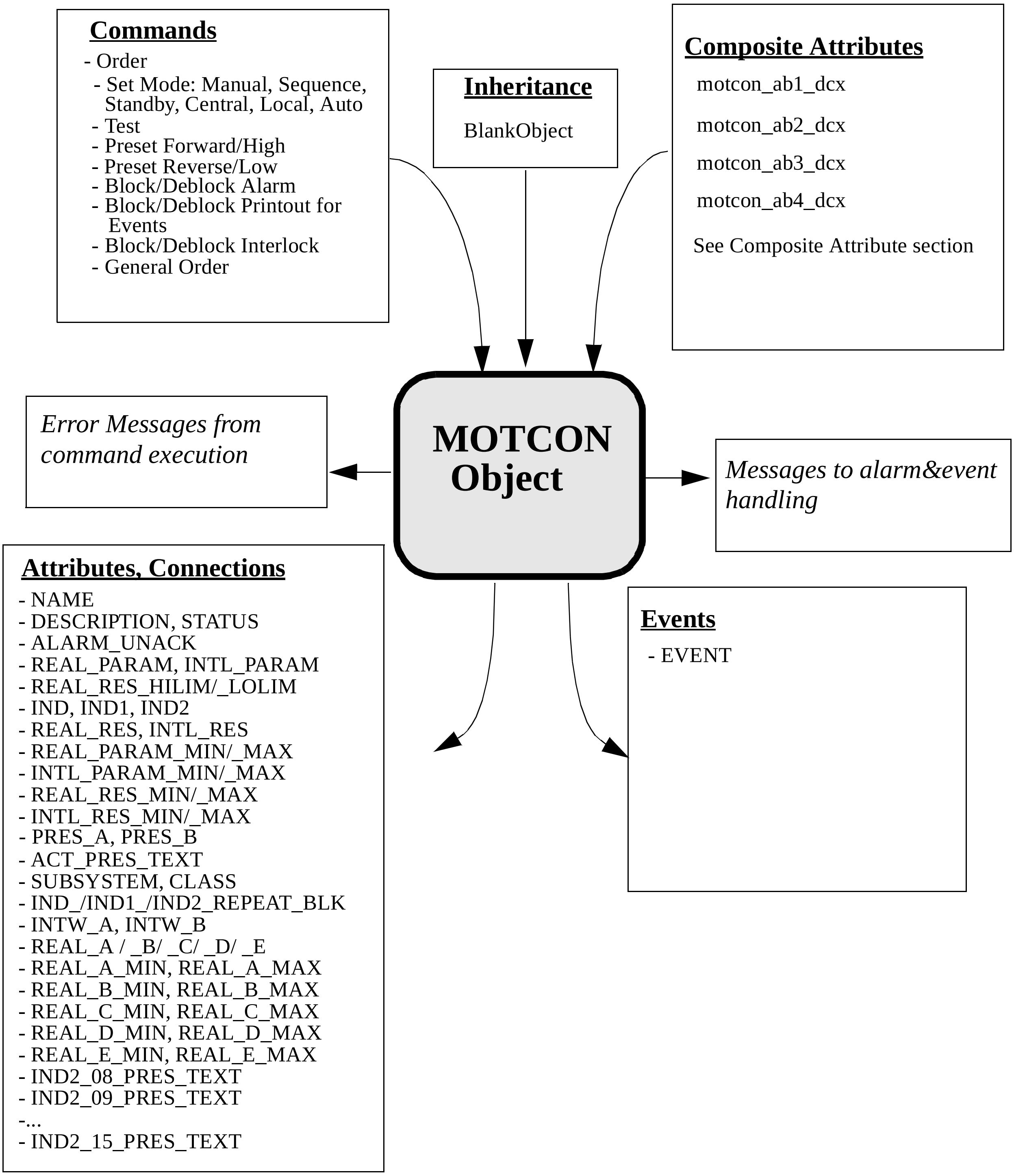


Figure 3-18. Overview of the Object Type MOTCON

3.14.3 Attributes

The object attributes are a subset of the corresponding attributes of the MOTCONs residing in the ABB Master Process Station. Any detailed information about an attribute can be collected from the ABB Master Database Element or Functional Units manuals

Table 3-103. MOTCON object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the object. See Table 3-104
ALARM_UNACK	alarm_unack	Alarm unacknowledged. See Table 3-99
REAL_PARAM	Float[8,2]	A real value, used by AMPL.
INTL_PARAM	IntLong	An integer value, used by AMPL.
REAL_RES_HILIM	Float[8,2]	A high limit value used by AMPL to make limit detection on a real value, for instance REAL_RES. Nominal current of motor.
REAL_RES_LOLIM	Float[8,2]	A low limit value used by AMPL to make limit detection on a real value, for instance REAL_RES.
IND	ind	See Table 3-105
IND1	IntShort	Indication 1 terminal.
IND2	IntShort	Indication 2 terminal.
REAL_RES	Float[8,2]	Motor current in percentage of rated current. A real value from AMPL.
INTL_RES	IntLong	An integer value from AMPL.
REAL_PARAM_MIN	Float[8,2]	Minimum value for REAL_PARAM.
REAL_PARAM_MAX	Float[8,2]	Maximum value for REAL_PARAM.
INTL_PARAM_MIN	IntLong	Minimum value for INTL_PARAM.
INTL_PARAM_MAX	IntLong	Maximum value for INTL_PARAM.
REAL_RES_MIN	Float[8,2]	Shows the lowest of the motor’s measure values in percentage of the motor’s rated current.
REAL_RES_MAX	Float[8,2]	Shows the highest of the motor’s measure values in percentage of the motor’s rated current.
INTL_RES_MIN	IntLong	Minimum value for INTL_RES.
INTL_RES_MAX	IntLong	Maximum value for INTL_RES.
PRES_A	String[11]	Presentation of position A.
PRES_B	String[11]	Presentation of position B.
ACT_PRES_TEXT	String[21]	Presentation of active interlocks. The string comes from one of the texts IND2_08_PRES_TEXT to IND2_14_PRES_TEXT. The text is prioritized by the active signal with the lowest number among the signals IND2_08 TO IND2_14.

Table 3-103. MOTCON object type attributes (Continued)

Attribute	OTDL Data Type	Description
SUBSYSTEM	IntSmall	Process section 0-16.
CLASS	IntSmall	Object class.
IND_REPEAT_BLK	ind_repeat_blk	See Table 3-96
IND1_REPEAT_BLK	IntShort	Shows which of the signals IND1_00 to IND1_15 that have repeated error blocks.
IND2_REPEAT_BLK	IntShort	Shows which of the signals IND2_00 to IND2_15 that have repeated error blocks.
INTW_A	intw_a	Additional integers. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically to the presentation interpreter. See Table 3-97
INTW_B	intw_b	Additional integers. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically to the presentation interpreter. See Table 3-98
REAL_A	Float[8,2]	Additional real. To be used by the AMPL-program that handles the object whenever so needed. These properties will be sent cyclically.
REAL_B	Float[8,2]	Additional real. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_C	Float[8,2]	Additional real. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_D	Float[8,2]	Additional real. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_E	Float[8,2]	Additional real. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically.
REAL_A_MIN	Float[8,2]	Minimum value for REAL_A.
REAL_A_MAX	Float[8,2]	Maximum value for REAL_A.
REAL_B_MIN	Float[8,2]	Minimum value for REAL_B.
REAL_B_MAX	Float[8,2]	Maximum value for REAL_B.
REAL_C_MIN	Float[8,2]	Minimum value for REAL_C.
REAL_C_MAX	Float[8,2]	Maximum value for REAL_C.
REAL_D_MIN	Float[8,2]	Minimum value for REAL_D.
REAL_D_MAX	Float[8,2]	Maximum value for REAL_D.
REAL_E_MIN	Float[8,2]	Minimum value for REAL_E.
REAL_E_MAX	Float[8,2]	Maximum value for REAL_E.
IND2_08_PRES_TEXT	String[21]	For presentation of C-interlock 1 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_08. Highest priority.

Table 3-103. MOTCON object type attributes (Continued)

Attribute	OTDL Data Type	Description
IND2_09_PRES_TEXT	String[21]	For presentation of C-interlock 2 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_09.
IND2_10_PRES_TEXT	String[21]	For presentation of B-interlock 3 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_10.
IND2_11_PRES_TEXT	String[21]	For presentation of B-interlock 4 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_11.
IND2_12_PRES_TEXT	String[21]	For presentation of B-interlock 1 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_12.
IND2_13_PRES_TEXT	String[21]	For presentation of B-interlock 2 in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_13.
IND2_14_PRES_TEXT	String[21]	For presentation of A-interlock in object display or as ACT_PRES_TEXT when the interlock is active. Associated with IND2_14. Lowest priority.
IND2_15_PRES_TEXT	String[21]	Not used.

Table 3-104. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
IMPLEMENTED	1	Implemented
ALARM_BLK	2	Alarm is blocked.
ALARM_PERIOD_BLK	3	Alarm period block.
PRINT_BLK	4	Printout is blocked.
IND1_DIST	5	IND1 disturbance. Controls the color priority at fails. True when at least one property within the block property IND1 is true.
IND2_DIST	6	IND2 disturbance. Controls the color priority at fails. True when at least one property within the block property IND2 is true.
SELECTED	7	Object selected by the operator.
AU_IND	8	Indicates whether there exists unacknowledged alarms. True when at least one bit in the property ALARM_UNACK is set.
REPEAT_BLK	9	Blocks repeated fails active/not active.
OH_SPARE_01	10	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_02	11	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_03	12	Spare property for future object handling functions. Will be sent together with the STATUS property.

Table 3-104. TdStatus. Status bits (Continued)

Bits in TdStatus	Bit number	Description
OH_SPARE_04	13	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_05	14	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_06	15	Spare property for future object handling functions. Will be sent together with the STATUS property.
OH_SPARE_07	16	Spare property for future object handling functions. Will be sent together with the STATUS property.
BOOL_A	17	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_B	18	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_C	19	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_D	20	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_E	21	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_F	22	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_G	23	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.
BOOL_H	24	Additional Boolean. To be used by the AMPL-program that handles the object whenever so needed. This property is sent cyclically, and together with the STATUS property.

Table 3-105. Ind. Status bits

Bits in ind	Bit number	Description
IND1_00	1	Control voltage absent/restored.
IND1_01	2	Bimetal relay has tripped/triggered.

Table 3-105. Ind. Status bits (Continued)

Bits in ind	Bit number	Description
IND1_02	3	Local stop.
IND1_03	4	Safety monitor tripped/triggered.
IND1_04	5	Main contactor error/error acknowledged.
IND1_05	6	Monitor Low tripped/restored.
IND1_06	7	Monitor High tripped/restored.
IND1_07	8	Position A reached/left.
IND1_08	9	Collective fault indication.
IND1_09	10	Collective unacknowledged fault indication.
IND1_10	11	Current limit 100% exceeded/re-entered.
IND1_11	12	Position B reached/left.
IND1_12	13	Run.
IND1_13	14	Preselected start order 1.
IND1_14	15	Preselected start order 2.
IND1_15	16	Exchange of start order in progress.
IND2_00	17	Blocked IB1 and IB3.
IND2_01	18	Control mode LOCAL on/off.
IND2_02	19	Control mode TEST on/off.
IND2_03	20	Control mode STANDBY on/off.
IND2_04	21	Control mode SEQUENCE on/off.
IND2_05	22	Operation mode AUTO on/off.
IND2_06	23	Started in AUTO.
IND2_07	24	Ready to run.
IND2_08	25	Interlock IC1 on/off.
IND2_09	26	Interlock IC2 on/off.
IND2_10	27	Interlock IB3 on/off.
IND2_11	28	Interlock IB4 on/off.
IND2_12	29	Interlock IB1 on/off.
IND2_13	30	Interlock IB2 on/off.
IND2_14	31	Interlock IA on/off.
IND2_15	32	Block. B-interlocking active.

3.14.4 Connections

The MOTCON object does not have any connections because they are only applicable for Object Types that are implemented as User Objects and for Basic Objects.

3.14.5 Composite Attributes

The attribute list behind the Composite Attribute name describes what attributes that are included.

motcon_ab1_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, ACT_PRES_TEXT, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E

motcon_ab2_dcx:

> STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E

motcon_ab3_dcx:

STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_PARAM, INTL_PARAM, REAL_RES_HILIM, REAL_RES_LOLIM, REAL_RES_MIN, REAL_RES_MAX, INTL_RES_MIN, INTL_RES_MAX, REAL_A_MIN, REAL_A_MAX, REAL_B_MIN, REAL_B_MAX, REAL_C_MIN, REAL_C_MAX, REAL_D_MIN, REAL_D_MAX, REAL_E_MIN, REAL_E_MAX, IND1_REPEAT_BLK, IND2_REPEAT_BLK, NAME, DESCRIPTION

motcon_ab4_dcx:

STATUS, IND_REPEAT_BLK, ALARM_UNACK, IND, REAL_RES, INTL_RES, INTW_A, INTW_B, REAL_A, REAL_B, REAL_C, REAL_D, REAL_E, PRES_A, PRES_B, ACT_PRES_TEXT, REAL_PARAM, INTL_PARAM, REAL_RES_HILIM, REAL_RES_LOLIM, REAL_RES_MIN, REAL_RES_MAX, INTL_RES_MIN, INTL_RES_MAX, REAL_A_MIN, REAL_A_MAX, REAL_B_MIN, REAL_B_MAX, REAL_C_MIN, REAL_C_MAX, REAL_D_MIN, REAL_D_MAX, REAL_E_MIN, REAL_E_MAX, IND1_REPEAT_BLK, IND2_REPEAT_BLK, NAME, DESCRIPTION, IND2_08_PRES_TEXT, IND2_09_PRES_TEXT, IND2_10_PRES_TEXT, IND2_11_PRES_TEXT, IND2_12_PRES_TEXT, IND2_13_PRES_TEXT, IND2_14_PRES_TEXT

3.14.6 Commands

The commands are treated in the same way as for calculated objects in ABB Master Process Stations. The table below shows the ORDER command, the corresponding operation parameters and the behavior of the object on such a command.

Table 3-106. MOTCON operation on the ORDER Command

Type of function for corresponding parameters	**Parameters to the ORDER command:** **Opcode** **OpProp**	**Functionality** **Output** **Exceptions**
MAN Set to manual mode	dcSET_DCX, dcRESET_DCX dcGENE06BOOL_DCX	The motor may be started directly from the selected point of control, provided the motor is ready for start. MORD_05 = 1.
AUTO Set to auto mode	dcSET_DCX, dcRESET_DCX dcGENE05BOOL_DCX	The process controls the motor by way of a AMPL program. This mode is e. g. used for on/off control of levels and for automatic start-up and shutdown of belt conveyors. MORD_04 = 1.
FORWARD/HIGH Preset forward/high	dcSET_DCX, dcRESET_DCX dcGENE09BOOL_DCX	MORD_08 = 1.
REVERSE/LOW Preset reverse/low	dcSET_DCX, dcRESET_DCX dcGENE10BOOL_DCX	MORD_09 = 1.

Table 3-106. MOTCON operation on the ORDER Command (Continued)

START Start motor	dcSET_DCX, dcRESET_DCX dcGENE11BOOL_DCX	Motor can be started when all interlocks are satisfied, there are no alarms and the point of control is not STANDBY. MORD_10 = 1.
STOP Stop motor	dcSET_DCX, dcRESET_DCX dcGENE12BOOL_DCX	Stops the motor. MORD_11 = 1.
Central Control Set to central point of control	dcSET_DCX, dcRESET_DCX dcGENE04BOOL_DCX	The motor is controlled from MasterView 800, AS 500 or Advant Controller 4xx. Control from other locations are blocked. MORD_03 = 1.
Sequence Control Set to sequence point of contro3l	dcSET_DCX, dcRESET_DCX dcGENE16BOOL_DCX	SEQUENCE is the default point of control. The motor is controlled from a superior AMPL program, e.g. for group start. Control from other locations are blocked. MORD_15 = 1.
Local Control Set to local point of control	dcSET_DCX, dcRESET_DCX dcGENE01BOOL_DCX	The motor is controlled from a local desk or panel. Control from other locations are blocked. MORD_00 = 1.
Test Control Set to test point of control	dcSET_DCX, dcRESET_DCX dcGENE02BOOL_DCX	The motor is controlled from a stop/start station adjacent to the motor. Is suitable for testing the motor in the field. The control modes MAN and AUTO are not available in point of control TEST. Control from other locations are blocked. MORD_01 = 1.
Out of Service Set to standby point of control	dcSET_DCX, dcRESET_DCX dcGENE03BOOL_DCX	The motor is stopped and all orders to it are blocked. MORD_02 = 1.
Alarm Block Blocks alarms handling for interlocks	dcBLOCK_DCX dcDIST_ALARM_DCX	If not already blocked, the STATUS bit ALARM_BLK is set to true.
Alarm Deblock Deblocks alarms handling for interlocks	dcDEBLOCK_DCX dcDIST_ALARM_DCX	If blocked the STATUS bit ALARM_BLK is set to false.
Print Block Blocks printout for events	dcBLOCK_DCX dcDIST_PRINT_DCX	If not already blocked, the STATUS bit PRINT_BLK is set to true.
Print Deblock Deblocks printout for events	dcDEBLOCK_DCX dcDIST_PRINT_DCX	If blocked the STATUS bit PRINT_BLK is set to false.
Interlock Block	dcBLOCK_DCX dcGENE07BOOL_DCX	All A and B interlockings are blocked when the TEST point of control is selected.

Table 3-106. MOTCON operation on the ORDER Command (Continued)

Interlock Activated	dcDEBLOCK_DCX dcGENE08BOOL_DCX	An interlock is active when the AMPL-input is 0 (false).
Order 1 General order 1	dcSET_DCX, dcRESET_DCX dcGENE13BOOL_DCX	MORD_12 = 1.
Order 2 General order 2	dcSET_DCX, dcRESET_DCX dcGENE14BOOL_DCX	MORD_13 = 1.
Order 3 General order 3	dcSET_DCX, dcRESET_DCX dcGENE15BOOL_DCX	MORD_14 = 1.

3.14.7 Events

The MOTCON object type has one event called EVENT.

Table 3-107. The event EVENT

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs. The event subscriptions of the MOTCON objects always request for at least the STATUS attribute to be returned.

3.15 Uninterruptable Power Supply - UPS

Uninterruptable Power Supply is an IMS hardware option. Installation of the UPS hardware is described in the *Advant Station Hardware User's Guide*. Make sure that you connect the UPS directly to the first RS-232 port of the Advant Station System Unit.

If your Advant Station IMS is delivered from ABB with all options installed there are no additional procedures to install the UPS. If the UPS option is purchased after delivery you have to install the corresponding UPS software as well. Read IMS User's Guide how to install the software. The name of the UPS software is IMS_UPS.

The UPS is configured for a connection towards the first RS-232 port. By default it is also configured to handle the UPS SV520 and SV521 respectively. These can only support one Advant Station at a time. The UPS software has the ability to support other UPS hardware making it possible to supply several Advant Stations at a time with Uninterruptable power. To configure and use such functionality you must contact ABB for instructions.

3.15.1 How to Use

The UPS Object Type provides functionality to supervise a UPS equipment. It can work in 3 different modes: Supervising, LocalEcho and DistEcho.

In all 3 modes it shuts down the Advant Station when battery is low and reports all status changes with local IMS error logs.

In supervising mode it supervises the UPS and informs all other UPS objects in other Advant Stations about any UPS status changes. It also informs all Advant Stations users about the changes by sending event messages (to the operator stations and local system message logs).

In DistEcho mode it takes care of shutdown when battery is low but will also send event messages to inform operators about all status changes. This mode should be used when an Advant Station is placed in another control network than the supervisor UPS object as event messages are distributed only within the own control network or within the plant network.

In LocalEcho mode the UPS only accepts all status messages, sends status changes to the system message log and shuts down the IMS when battery is low:

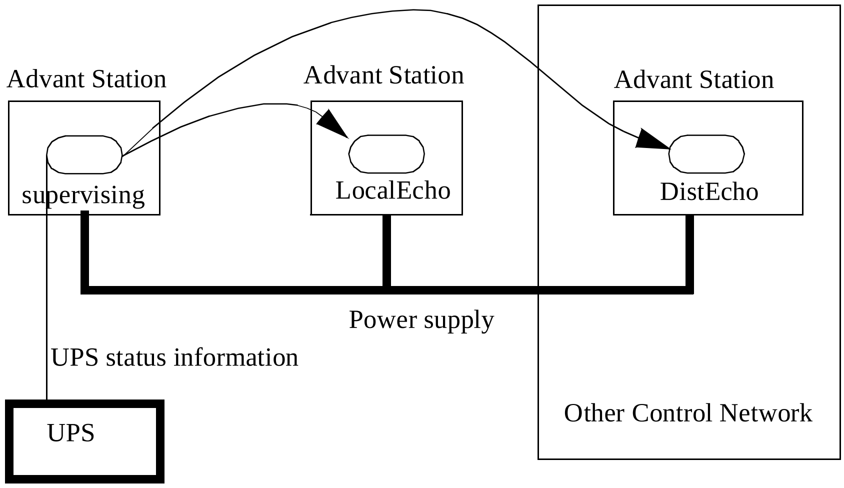


Figure 3-19. One UPS object can supervise several Advant Stations using other UPS objects to manage the other Advant Nodes.

3.15.2 Overview

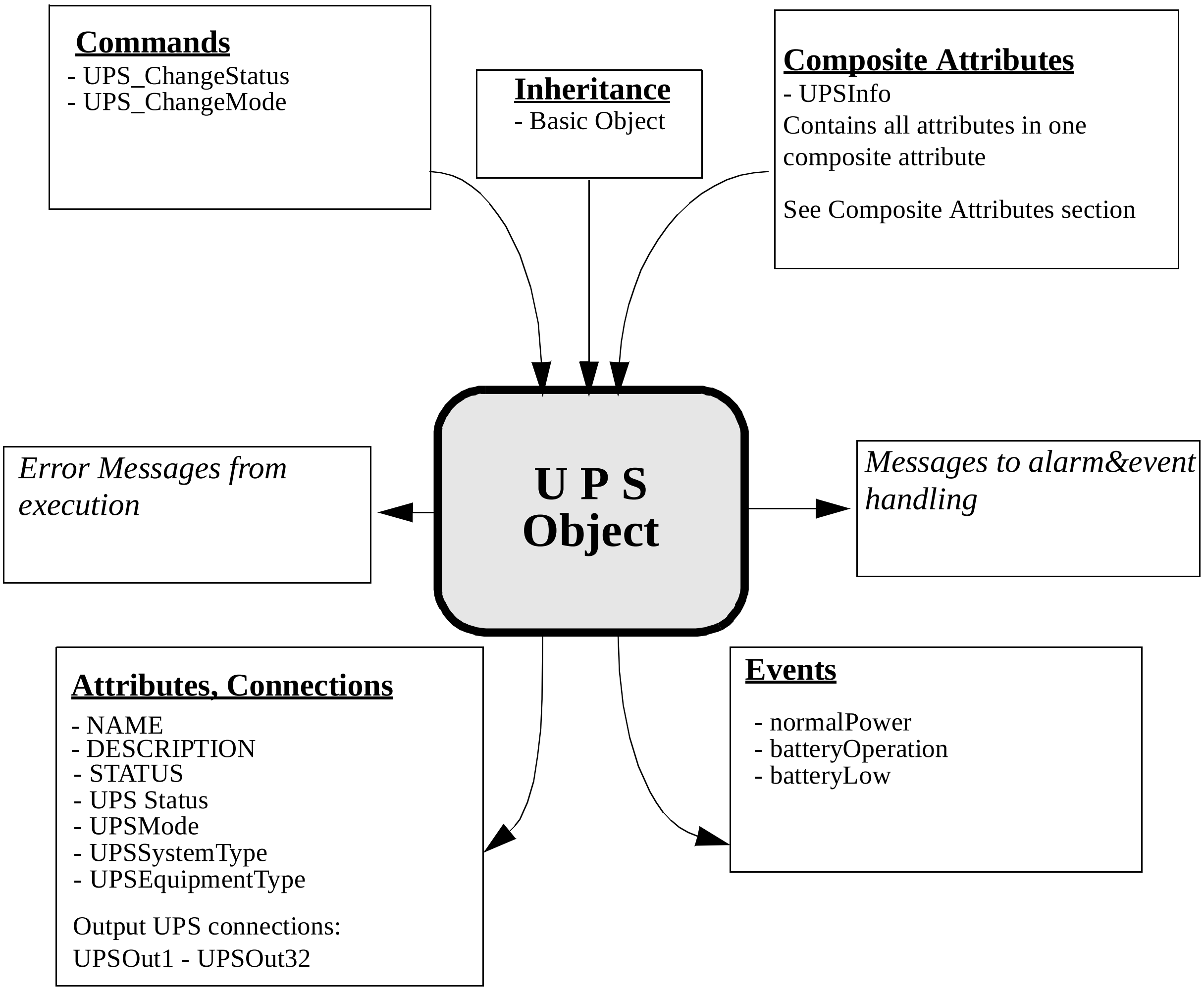


Figure 3-20. Overview of the Object Type UPS

3.15.3 Attributes

The object attributes are few and require little explanation. The most important information is the STATUS, the UPSStatus and the UPSMode attributes.

To be able to find other UPS objects, all UPS objects shall be named UPS_<Node_Number>

Table 3-108. UPS object type attributes

Attribute	OTDL Data Type	Description - Possible statuses
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the object, see Table 3-109
UPSStatus	IntSmall	
UPSMode	IntSmall	The object can be configured to execute in three different ways: UPS_M_Supervise = 1 The object supervises the UPS and informs all other UPS objects about the current state. The supervision also means that events are sent when status changes occur. UPS_M_LocalEcho = 2 The object updates status and sends system message logs when commands are received. In addition, it shuts the system down when BatteryLow is received. UPS_M_DistEcho = 3 The object updates status, sends system message logs and sends event messages when commands are received. In addition, it shuts the system down when BatteryLow is received. UPS_M_NotActive = 0 UPS is not connected to this Advant Station.

Table 3-108. UPS object type attributes (Continued)

Attribute	OTDL Data Type	Description - Possible statuses
UPSSystemType	IntSmall	The object can be configured to operate in two different systems: UPS_S_Master = 0 The UPS operates in a ABB Master network. UPS_S_MOD300 = 1 The UPS operates in a MOD300 network.
UPSEquipmentType	IntSmall	The connected UPS equipment can be one of the following types: UPS_E_NONE = 0 No UPS equipment is connected. UPS_E_FISKARS = 1 FISKARS UPS equipment is used. UPS_E_MP51 = 2 MP51 equipment is used.

Table 3-109. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run.
error	2	Indicates the execution status of the object. If True, the objects execution is disturbed, indicating an internal object error.
updated	3	Indicates if the object has been updated since object execution started.

3.15.4 Connections

The UPS has 32 optional Output Object Connections. All of these connections are configured for the supervisor objects but not for UPS objects running in other modes. Through the connections the supervisor object updates the UPS objects in LocalEcho and DistributedEcho mode.

Table 3-110. Output connection

Name	Direction	Type/Format	Description
UPSOut1 - UPSOut32	Output	Object	Optional object connection. Used as output from the Supervisor object to the Echo objects.

3.15.5 Composite Attributes

UPSInfo:

NAME, DESCRIPTION, STATUS, UPSStatus, UPSMode, UPSSystemType.
Makes it possible to read info from UPS object using only one attribute.

3.15.6 Commands

The command UPS_ChangeStatus is executed synchronously, i.e. it waits for the other UPS object to receive the command before it continues. The command has the following syntax:

UPS_ChangeStatus(NewStatus)

Table 3-111. UPS_ChangeStatus Command

Value of NewStatus	**New Status**
UPS_S_OK = 0	OK: Normal power supply.
UPS_S_Battery = 1	Battery: Main power is lost.
UPS_S_BatteryLow = 2	Battery Low: Shut down as fast as possible.
UPS_S_Off = -1	UPS is not working or no UPS connected.
UPS_S_NotOnLine = -2	Error: UPS is not On Line.
UPS_S_ProgramError = -3	Error: Access to UPS failed.
UPS_S_LineError = -4	Error: UPS device does not exist or failure to open the device. Check your installation.

The command UPS_ChangeMode is included for future use. It is not used by the AS 500 products in this release. It has the following syntax:

UPS_ChangeMode (New Mode)

Table 3-112. UPS_ChangeMode Command

Value of New Mode	**New Mode**
UPS_M_NotActive = 0	UPS is not active in this Advant Station.
UPS_M_Supervise = 1	The object supervises the UPS and informs all other UPS objects about the current state. The supervision also means that events are sent when status changes occur.
UPS_M_LocalEco = 2	The object updates status and sends system message log when commands are received. In addition, it shuts the system down when BatteryLow is received.
UPS_M_DistEcho = 3	The object updates status, sends system message logs and sends event messages when commands are received. In addition, it shuts the system down when BatteryLow is received.

3.15.7 Events

The UPS object type has one event called UPS.

Table 3-113. The event EVENT

Name	Value/explanation
normalPower	The event is raised when a transition from BatteryOperation to NormalPower occurs.
batteryOperation	The event is raised when a transition from Normal Power to BatteryOperation occurs.
batteryLow	The event is raised when BatteryLow occurs.

3.16 coExecutor Object

3.16.1 How to Use

The coExecutor Object exists as a Basic Object type in AdvaInform Object Handling.

It executes any software using ordinary posix calls. The name and arguments to the software is configurable through the command changeExecString. The coExecutor verifies that the software to execute exists. If the execution isn't terminated, and provided that the errorReport attribute is true, an error log is sent.

3.16.2 Overview

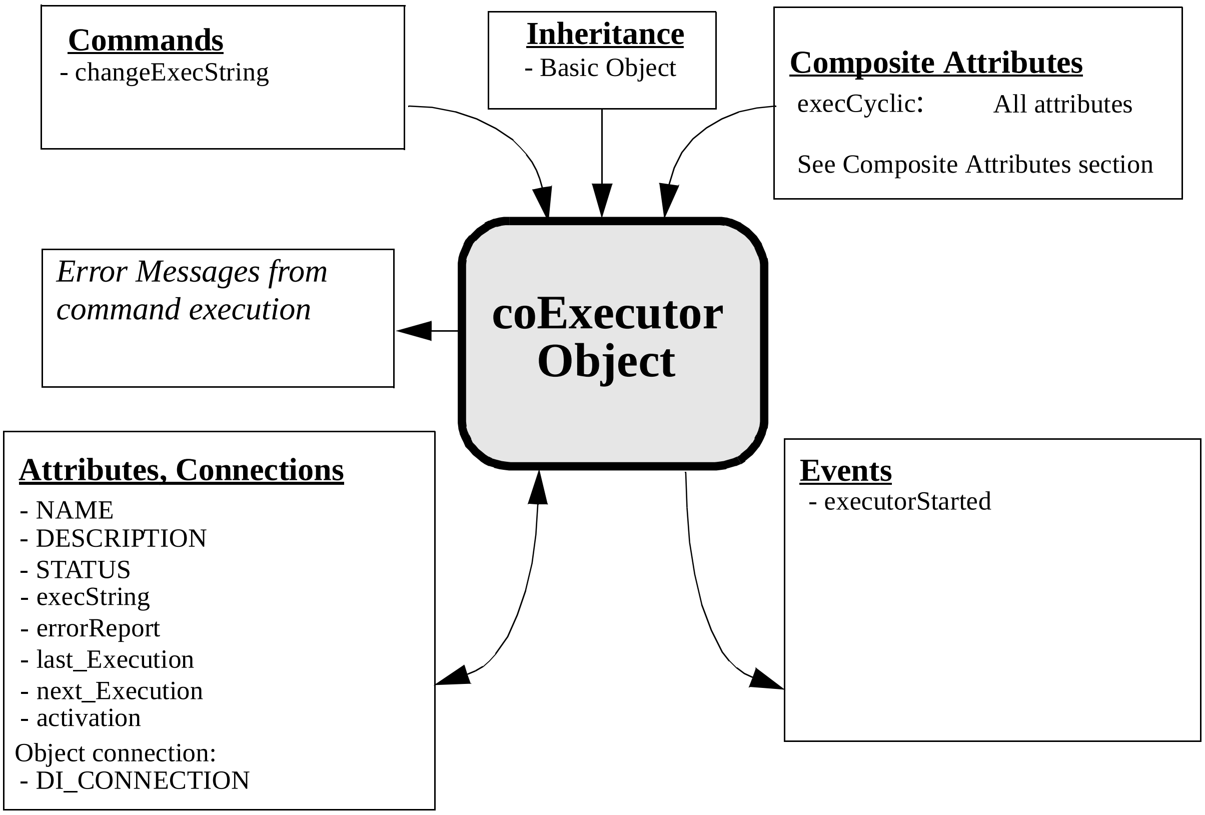


Figure 3-21. Overview of the coExecutor Object

3.16.3 Attributes

Table 3-114. coExecutor Object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	Td Status	Represents the status of the object. See Table 3-115
execString	String[255]	String containing filename and arguments to the software that will be executed.
errorReport	Boolean	Error reporting attribute. If the attribute is set to true an error text will be sent to the errorlog in case of failure of the execution start-up.
last_Execution	String[25]	The time for the last execution.
next_Execution	String[25]	The time for the next execution.
activation	IntSmall	Configurable attribute which can be set by the operator. The attribute is used to control the triggered activation. See Table 3-116

Table 3-115. TdStatus. Status bits

Bits in TdSatus	Bit number	Description
active	1	Object active i.e. ready to run.
error	2	Indicates the execution status of the object. If True, the objects execution is disturbed, indicating an internal object error.
updated	3	Indicates if the object has been updated since object execution started.
ecexFailed	8	This bit is default False. If an attempt to execute the software pointed at fails, the bit is set to True. The bit is reset when the object executes correctly next time.

Table 3-116. activation attribute

Value	Explanation
0	The execution will be performed only if the DI changes value from 0 to 1.
1	The execution will be performed only if the DI changes value from 1 to 0.
2	The execution will be performed whenever the DI changes value. (0 to 1, 1 to 0)
3	The execution will be performed whenever the DI toggles.(0 to 0, 0 to 1, 1 to 0, 1 to 1)

3.16.4 Connection

The coExecutor object consists of one optional DI-connection which makes it possible to execute on event.

Table 3-117. Input connection

Name	Direction	Type/Format	Description
DI_CONNECTION	Input	Object/DI	Optional DI input connection. Used to activate the object when the DI generates an event and the DI changes value (depending on the value of the activation attribute).

3.16.5 Composite Attribute

The coExecutor Object has one Composite Attribute:

execCyclic:

STATUS, execString, errorReport, last_ Execution, next_ Execution, activation

3.16.6 Commands

The coExecutor object has the following non inherited commands:

changeExecString:

With changeExecString it is possible to change the software to be executed. The syntax is:

– software argument1 argument2 ... argument10

The number of argument is optional from 0 to 10 and the software can be given an absolute path. If only the software name is given the coExecutor uses the environment variable PATH to find the software. For example:

– /user/username/a.out -l -more (absolute path to software “a.out” which have two arguments)

– ls -al (automatic search for software “ls” which have one argument)

normalOperation:

Executes the object which executes the software in the execString.

Table 3-118. Parameter to the command changeExecString.

Parameter	Direction	OTDL- and C-Format	Value/ explanation
changeString	Input	String[255] - OmfChar[255]	The file name of the software to execute and its arguments.

3.16.7 Event

The coExecutor object has one event called executorStarted.

Table 3-119. The event executorStarted

Name	Value/explanation
executorStarted	The event is raised when an object has started the software with success.

3.17 coBasicCalc Object

3.17.1 How to Use

The coBasicCalc Object exists as a Basic Object type in AdvaInform Object Handling.

The object processes the input which consists of 1 to 10 floating point values. It can be configured to perform one of a number of different calculations on the input.

The object consists of the following:

- The attributes are inherited from the AI object type. This means that the attributes are accessible from all Advant and Master functions that support AI objects.
- It has the general behavior of an AI making it possible to interact with and present on an operator station or on AdvaSoft for Windows. The object can be presented using any of the existing AI display elements. The Object Displays can also be used.
- It has 1 to 10 optional floating point inputs (so called single connections).
- It has one optional DI as input making it possible to execute on event.
- It provides a number of predefined algorithms. The result is stored in the VALUE and STATUS attributes.
- It is configurable from the Object Handling Object Configuration tool.
- After each calculation the result is compared with the configured alarm limits and ranges. Limit transgressions result in events (or alarms, depending on how you configured it). They also result in changes of the STATUS attribute, DISTURBANCE and the limit status bit(s) are set. Range transgressions result in the setting of the status bits AIOVERFLOW and DISTURBANCE.
- For the user at an operator station the object behaves like any other calculated AI object. The calculation algorithm could be compared with a AMPL sequence in a MasterPiece.
- The result can be logged into numerical logs in AdvaInform History. All attributes can be read through the User API or by another User Object.

3.17.2 Overview

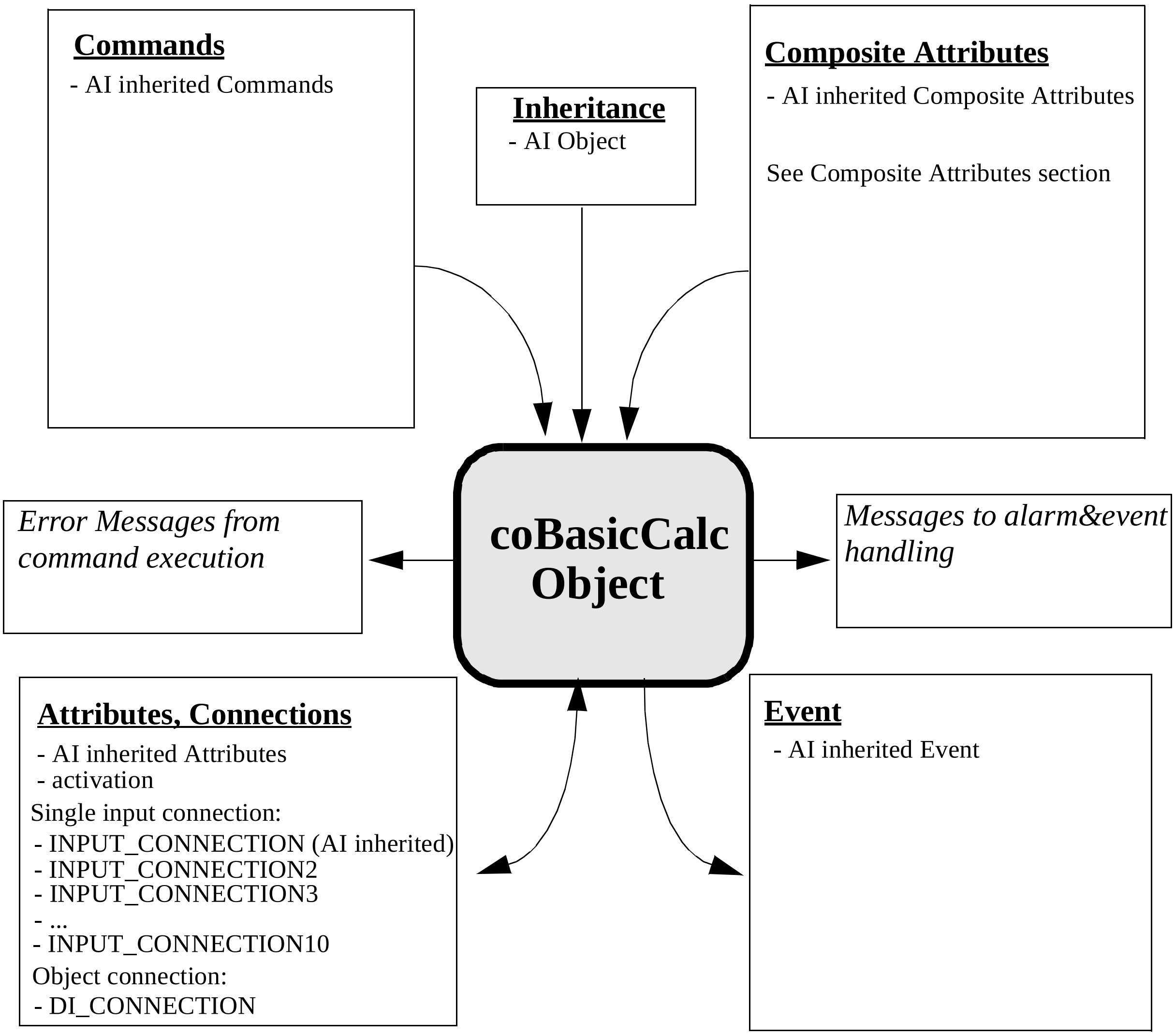


Figure 3-22. Overview of the coBasicCalc Object Type

3.17.3 Attributes

Attributes are inherited from both the Basic Object Type and the AI Object Type.

Table 3-120. coBasicCalc Object Type not inherited attribute.

Attribute	OTDL Data Type	Description
activation	IntSmall	Configurable attribute which can be set by the operator.The attribute is used to control the triggered activation. See Table 3-122

Table 3-121. Additional status bit in STATUS.

Bit in STATUS	Bit number	Description
AIOVERFLOW	16	This bit is set to true if the Range value (Max or Min) is transgressed.

Table 3-122. activation attribute

Value	Explanation
0	The calculation will be performed only if the DI changes value from 0 to 1.
1	The calculation will be performed only if the DI changes value from 1 to 0.
2	The calculation will be performed whenever the DI changes value. (0 to 1, 1 to 0)
3	The calculation will be performed whenever the DI toggles. (0 to 0, 0 to 1, 1 to 0, 1 to 1)

3.17.4 Connections

The coBasicCalc object has one inherited optional single connection from the AI object called INPUT_CONNECTION and 9 non inherited connections (INPUT_CONNECTION2-10). The inherited INPUT_CONNECTION has to be configured if the calculation is to take place.

It also has one non inherited DI_CONNECTION which makes it possible to execute on Event.

Table 3-123. Input connection

Name	Direction	Type/Format	Description
INPUT_CONNECTION INPUT_CONNECTION2 INPUT_CONNECTION3 ... INPUT_CONNECTION10	Input	Single/Float	Optional single input connections
DI_CONNECTION	Input	Object/DI	Optional DI connection. Used to activate the object when the DI generates an event and the DI changes value (depending on the value of the Activation attribute).

3.17.5 Composite Attributes

The composite attributes are inherited from the AI Object Type.

3.17.6 Commands

The commands are inherited from the AI Object Type.

3.17.7 Normal Operations

coBasicCalc has the following normal operations.

Table 3-124. coBasicCalc normal operations

Name	Value/explanation
CalcAverage_normop	Result = (INPUT_CONNECTION + ... + INPUT_CONNECTION*n) / n;* (1)
CalcSum_normop	Result = INPUT_CONNECTION + ... + INPUT_CONNECTION*n;* (1)
CalcSub_normop	Result = INPUT_CONNECTION - ... - INPUT_CONNECTION*n;*(1)
CalcMulti_normop	Result = INPUT_CONNECTION * ... * INPUT_CONNECTION*n;*(1)
CalcDivision_normop	Result = INPUT_CONNECTION / ... / INPUT_CONNECTION*n;*(1)
CalcMax_normop	Find the highest value of the configured input connections.
CalcMin_normop	Find the lowest value of the input configured connections.

(1) n <= 10.

3.17.8 Event

The event is inherited from the AI Object Type.

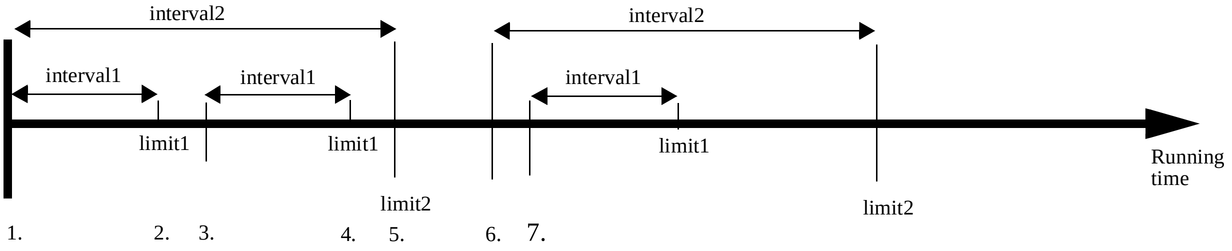
3.18 coRunTimeMeasurement Object

3.18.1 How to Use

The coRTM Object exists as a Basic Object type in AdvaInform Object Handling. The object will perform run time measurements and store the results into its own attributes.

The object has one mandatory DI input connection which is the signal for start and stop. The operation time for the object is accumulated and compared towards the alarm limits each time the object is stopped or, if configured, is executed cyclic. In case of transgression an event message to the event and alarm system is generated. An object event is also generated.

The coRunTimeMeasurement object has the following behavior if both limit1 and limit2 are in use:



1. When the object is triggered to start, the limit1 and limit2 are calculated according to limitInterval 1 and 2.
2. When the objects running time reaches limit1 an alarm is generated and limit1 status bit is set to true.
3. The operator acknowledges the limit1 alarm and the status bit is reset. The new limit1 time is calculated.
   – New limit1 = Current Run Time + limitInterval1;
4. When the objects running time reaches the “new” limit1 an alarm is generated and limit1 status bit is set to true.
5. When the objects running time reaches limit2 another alarm is generated and limit2 status bit is set to true.
6. The operator acknowledges the limit2 alarm and the corresponding status bit is reset. New values for limit2 are calculated from the time of the acknowledge.
7. The operator acknowledges the limit1 alarm. See 6.

3.18.2 Overview

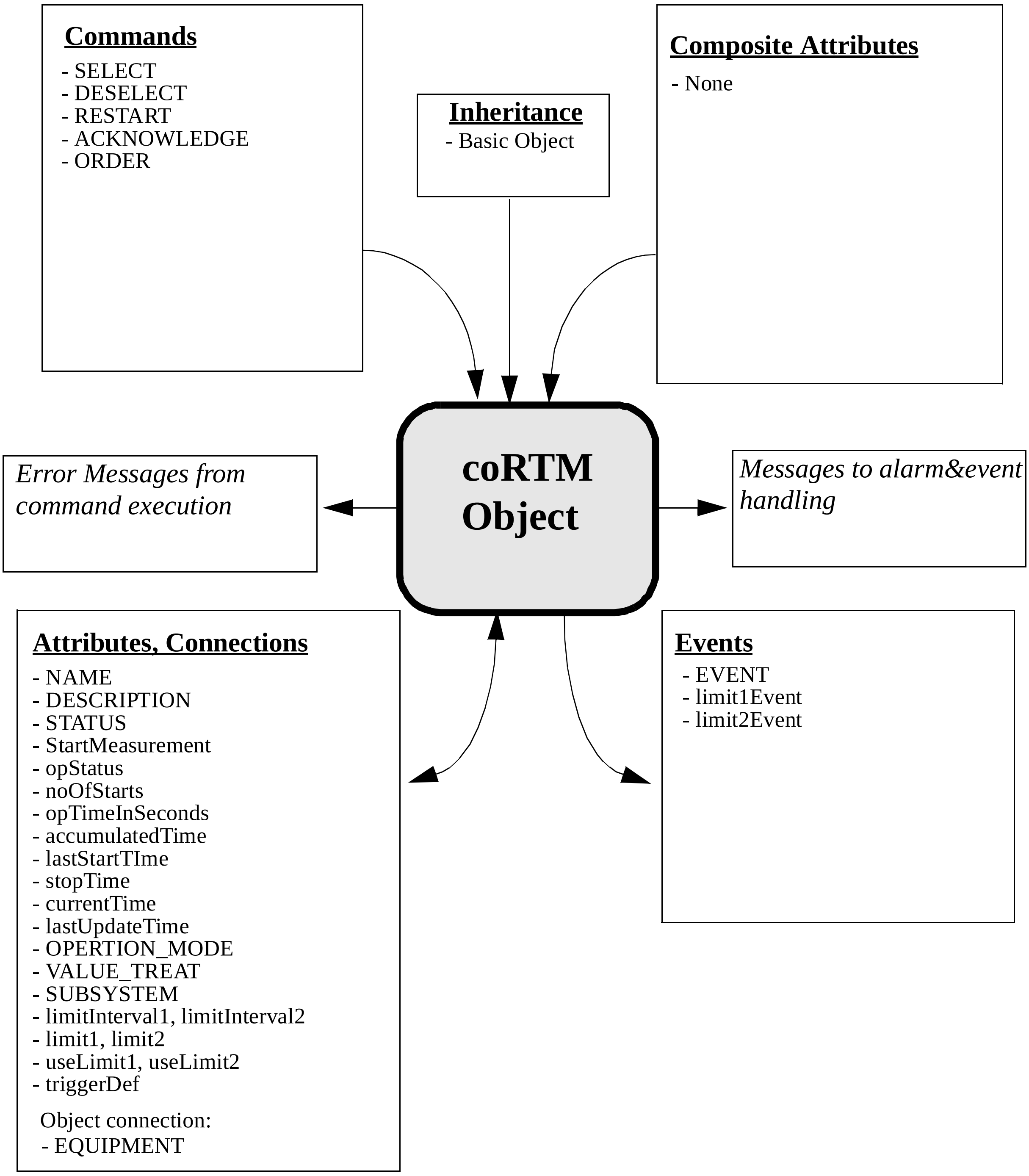


Figure 3-23. Overview of the coRTM Object type

3.18.3 Attributes

Table 3-125. coRTM Object Type attributes.

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the object, see Table 3-126
startMeasurement	IntLong	Start time for measurement.
opStatus	IntSmall	Equipment is running.
noOfStarts	IntLong	Number of starts for the object.
opTimeInSeconds	IntLong	Total accumulated run time in seconds.
accumulatedTime	IntLong	Accumulated run time in seconds since last start.
lastStartTime	IntLong	Last start time.
stopTime	IntLong	Last stop time.
currentTime	IntLong	Current measure time.
lastUpdateTime	IntLong	Last measure time.
OPERATION_MODE	IntShort	
VALUE_TREAT	IntShort	Value of event treatment definitions in MV/OS.
SUBSYSTEM	IntSmall	Process section 0 - 16.
limitInterval1	IntLong	Interval, in seconds between maintenance one.
limitInterval2	IntLong	Interval, in seconds between maintenance two.
limit1	IntLong	Alarm limit value 1. Compared to accumulated time.
limit2	IntLong	Alarm limit value 2. Compared to accumulated time.
useLimit1	Boolean	Alarm Limit1 used.
useLimit2	Boolean	Alarm Limit2 used.
triggerDef	Boolean	Configurable attribute to define how the Di strike will trigger the object. Value = false starts the measurement in case of DI strike “1” and stops it in case of DI strike “0”. Value = true is the opposite.

Table 3-126. TdStatus. Status bit.

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run.
error	2	Indicates the execution status of the object. If True, the objects execution is disturbed, indicating an internal object error.
updated	3	Indicates if the object has been updated since object execution started.
SELECTED	4	Indicates if the object is selected.
AUTO	5	
OPSTATUS	6	This bit is set if the device is in operation.
LIMIT1_TRANSGRESS	7	This bit is set to true when the limit1 is transgressed and the limit1Event is generated.
LIMIT2_TRANSGRESS	8	This bit is set to true when the limit2 is transgressed and the limit2Event is generated.

3.18.4 Connection

Table 3-127. Input connection.

Name	Direction	Type/Format	Description
EQUIPMENT	Input	Object/DI	The connection is used to activate the object when the DI toggles, provided that the value has changed to start position.

3.18.5 Commands

The coRTM object type has the following commands:

SELECT

With this command you can select the object.

DESELECT

With this command you can deselect the selected object.

RESTART

Command to restart the object.

ORDER

Sets the LIMIT1_TRANSGRESS and/or LIMIT2_TRANSGRESS status bit.

ACKNOWLEDGE

This command acknowledges alarm and resets the status bits. It also recalculates the alarm limit according to the limit interval.

NORMAL OPERATION

Accumulates the operation time for the object and registers the total number of starts and stops. Compares the accumulated time towards alarm limits (provided that the limits are in use). Generates events where the limits are transgressed. Also generates a message to the event and alarm system

3.18.6 Events

The coRunTimeMeasurement object has three events.

Table 3-128. The coRTM *objects event.*

Name	Value/explanation
EVENT	The event is raised as soon as any change of the value or the status of the object instance occurs.
limit1Event	This event is generated when Limit1 is transgressed.
limit2Event	This event is generated when Limit2 is transgressed.

Chapter 4 System Application Objects

4.1 Object Handler Supervisor Object - Daemon

4.1.1 How to Use

The object type Daemon incorporates functions for start/stop and supervision of Object Handlers, that are objects of type ObjectHandlerType. It also provides status information on all Object Handlers in the system. One Daemon exists in each IMS Station.

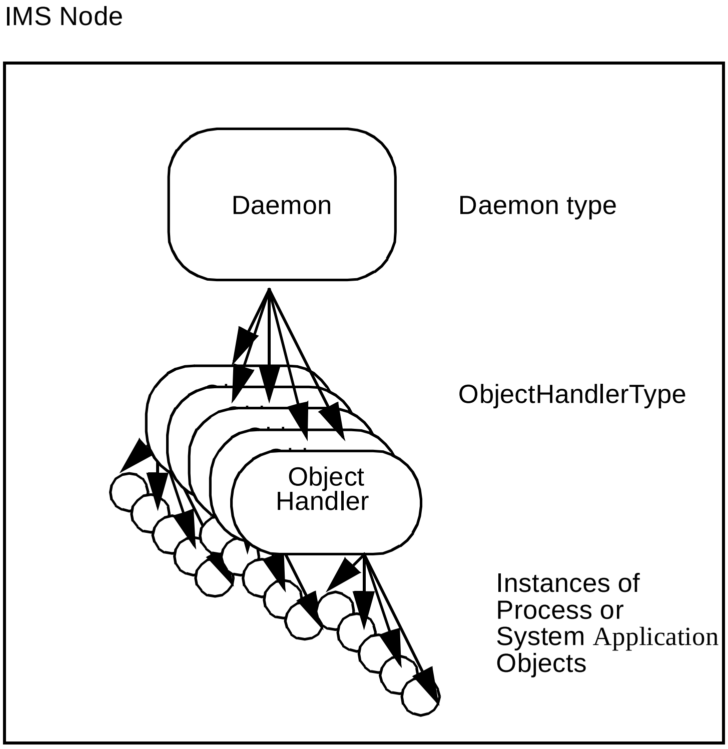


Figure 4-1. Daemon in an IMS Station

NOTE

It is strongly recommended for application programmers not to use other commands than OhSupervise and OhSumStatus and in particular not on Object Handlers not directly related to their applications.

4.1.2 Overview

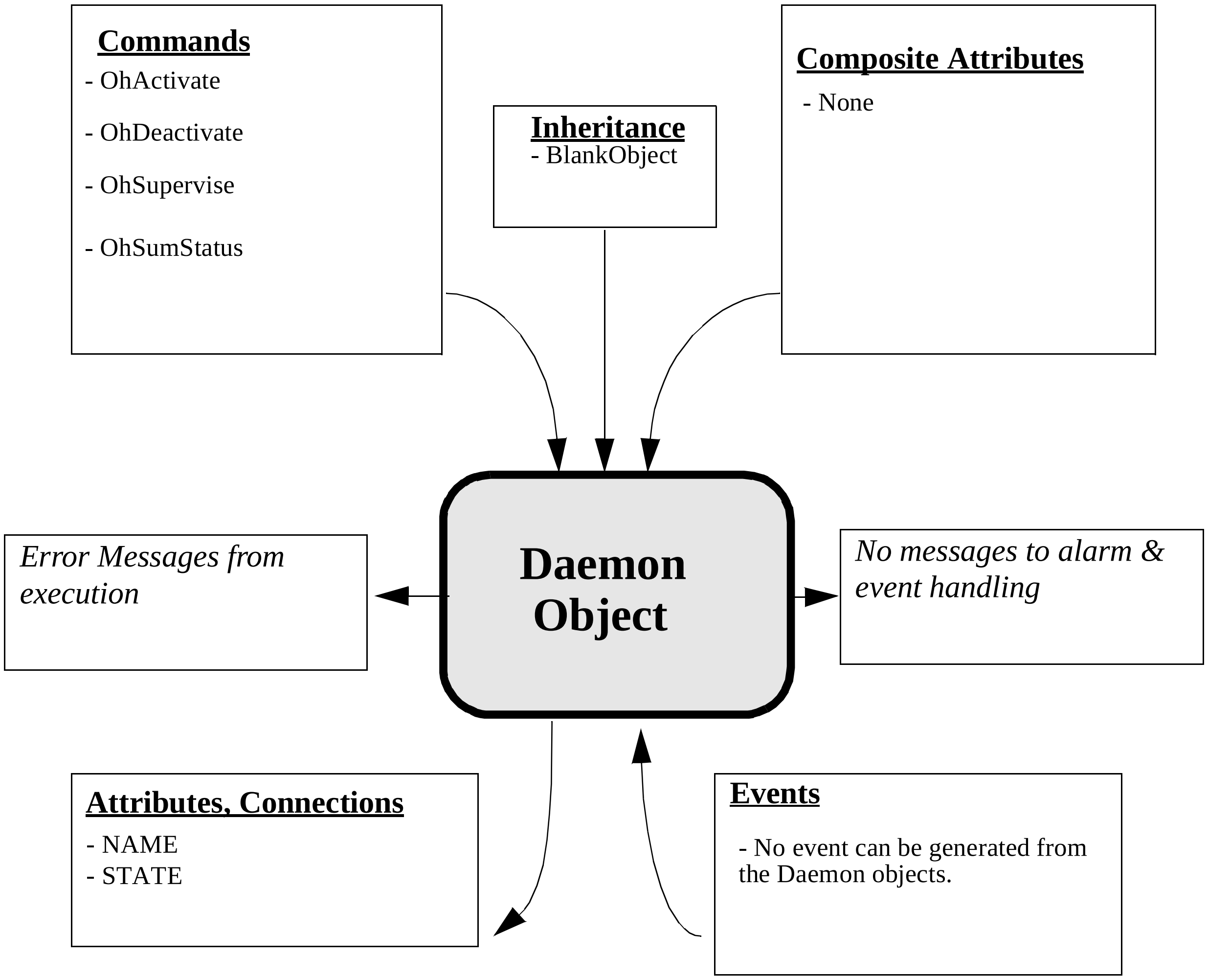


Figure 4-2. Overview of the Daemon

4.1.3 Attributes

Table 4-1. Daemon object type attributes

Attribute	OTDL -and C-Format	Description
NAME	TdName	Name of the daemon is always “$CODAEMON”.
STATE	IntShort - OmfShort	state = 0 : Started, Status OK state <> 0 : Error

4.1.4 Connections

The Daemon has no connections.

4.1.5 Composite Attributes

The Daemon has no Composite Attributes.

4.1.6 Commands

The Daemon has the following commands:

OhActivate

With **OhActivate** you start a dormant object handlers execution.

OhDeactivate

With **OhDeactivate** you **stop** an object handler. It will be dormant and can’t be accessed before started again with **OhActivate**. All object instances executed through that Object Handler will also be dormant.

OhSupervise

With **OhSupervise** you ask for the status of an object handler. See the return parameter for possible statuses.

OhSumStatus

With **OhSumStatus** you ask for the summary status of all object handlers. See the return parameter for possible statuses.

Syntax

OhActivate	(objectHandlerName, success)
OhDeactivate	(objectHandlerName, success)
OhSupervise	(objectHandlerName, status)
OhSumStatus	(sumStatus)

Table 4-2. Parameters to the Commands

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectHandlerName	input	TdName - gcString	The name of the object handler
success	output	IntShort - OmfShort	Indicates whether the operation was successful or not. For OhActivate: gcTrue, The ObjectHandler was successfully activated. gcFalse, The activation of the object handler failed. For OhDeactivate: gcTrue, The ObjectHandler was successfully deactivated. gcFalse, The deactivation of the object handler failed.
sumStatus	output	IntShort - OmfShort	The current summary state of the Object Handlers in the IMS node: gcTrue, all active object handlers are executing. gcFalse, one or several object handlers marked active are not executing.
status	output	IntShort - OmfShort	The state of an object handler process. gcTrue - The object handler is executing. gcFalse - The object handler is not executing when supposed to.

4.1.7 Events

The Daemon has no events.

4.2 ObjectHandler Object - ObjectHandlerType

4.2.1 How to Use

The object type ObjectHandlerType incorporates functions for Creating, Executing and Stopping Object Instances of the object types assigned to it.

Each object handler is given a unique object name and a number of object types assigned to it. For each of those object types it handles the run time execution of all object instances.

4.2.2 Overview

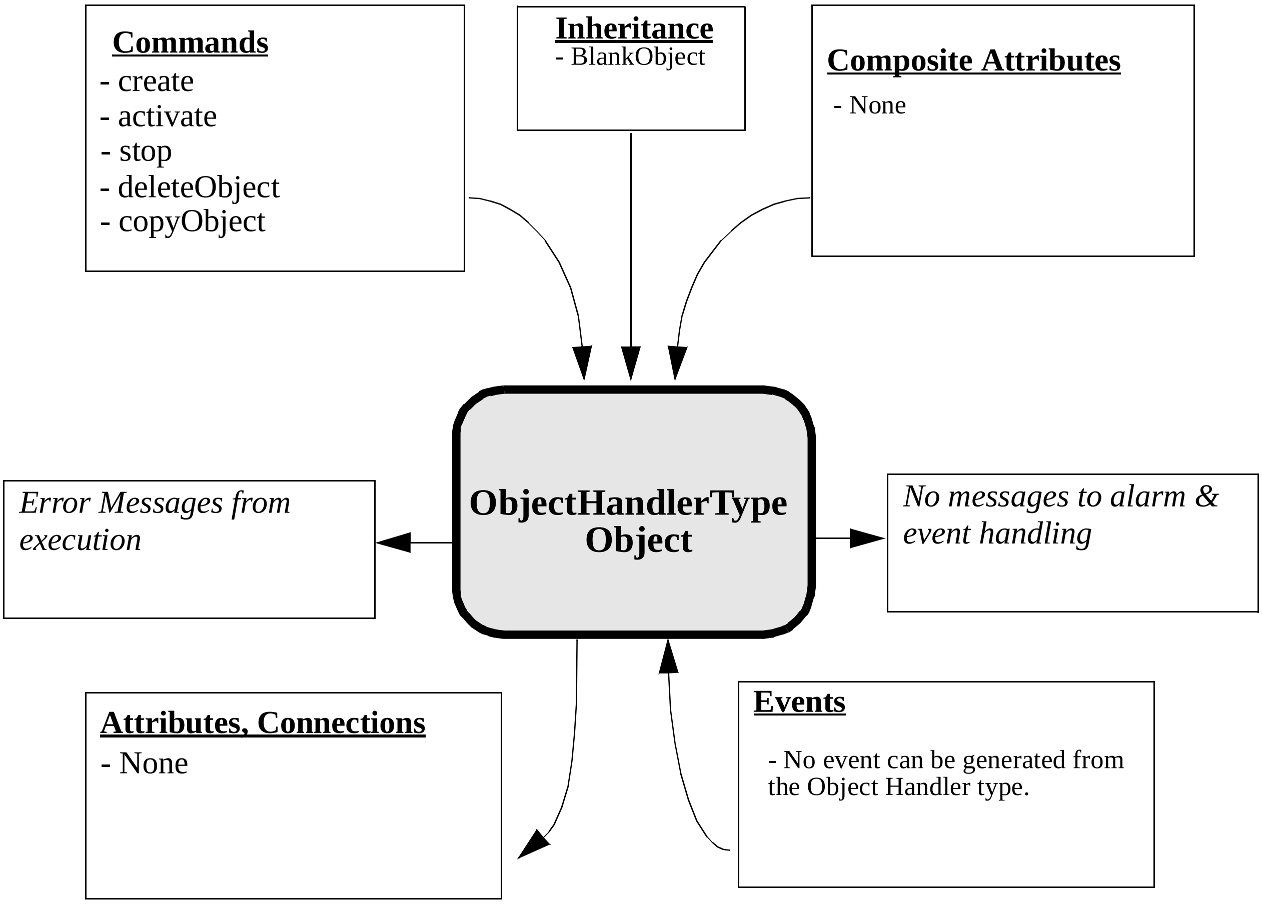


Figure 4-3. Overview of the ObjectHandlerType

4.2.3 Attributes

The ObjectHandlerType has no attributes.

4.2.4 Connections

The ObjectHandlerType has no connections.

4.2.5 Composite Attributes

The ObjectHandlerType has no Composite Attributes.

4.2.6 Commands

The ObjectHandlerType has the following commands:

create

> With **create** you create a new object. Just supply the object type, the name of the new object instance, a description and a normal operation number, normally 1, and the object is created. All attributes and connections are filled in with appropriate defaults. Cyclic execution is disabled.

activate

> With **activate** you start a dormant objects execution.

stop

> With **stop** finally you stop an object handler. It will be dormant and can’t be accessed before started again with **activate**.

deleteObject

> With **deleteObject** you delete an Object of a specified type. The Object may be active or inactive.

copyObject

> With **copyObject** you copy an object.

Syntax

```
create      (typeName, objectName, description, normalOperation)
activate    (typeName, objectName)
stop        ()
deleteObject(typeName, objectName)
copyObject  (oldObjectName, newObjectName)
```

Table 4-3. Parameters to the Command create

Parameter	Direction	OTDL- and C-Format	Value/explanation
typeName	input	TdName - TdName	The name of the object type
objectName	input	TdName - TdName	The name of the object instance to create
description	input	TdDescription-TdDescription	28 character description of the object
normalOperation	input	IntSmall - OmfSmall	Normal Operation to use, 1 to 10.

Table 4-4. Parameters to the Command activate

Parameter	Direction	OTDL- and C-Format	Value/explanation
typeName	input	TdName - gcString	The name of the object type
objectName	input	TdName - gcString	The name of the object instance to create

4.2.7 Events

The ObjectHandlerType has no events.

4.2.8 Object Handler Names

The following Object Handlers exist in the IMS software modules:

Table 4-5. Object Handler names

Software	Object Handler name	Description
AdvaInform Object Handling	basichandler	Handles the instances of the Basic Object types AI, AO, DI, DO, DAT, TXT
AdvaInform Calculations	calchandler	Handles the instances of the Calculation Object Type: CalcObject
AdvaInform Reports	reporthandler	Handles the instances of the Report Object Type: ReportObject

4.3 Event & Alarm Distributor Object - EADISTR

4.3.1 How to Use

The object EADISTR distributes events and event messages from other objects (applications) in IMS to all subscribers throughout the ABB Automation System. The event messages are sent to the EADISTR object as the commands distributeEvent and distributeSPCEvent.

4.3.2 Overview

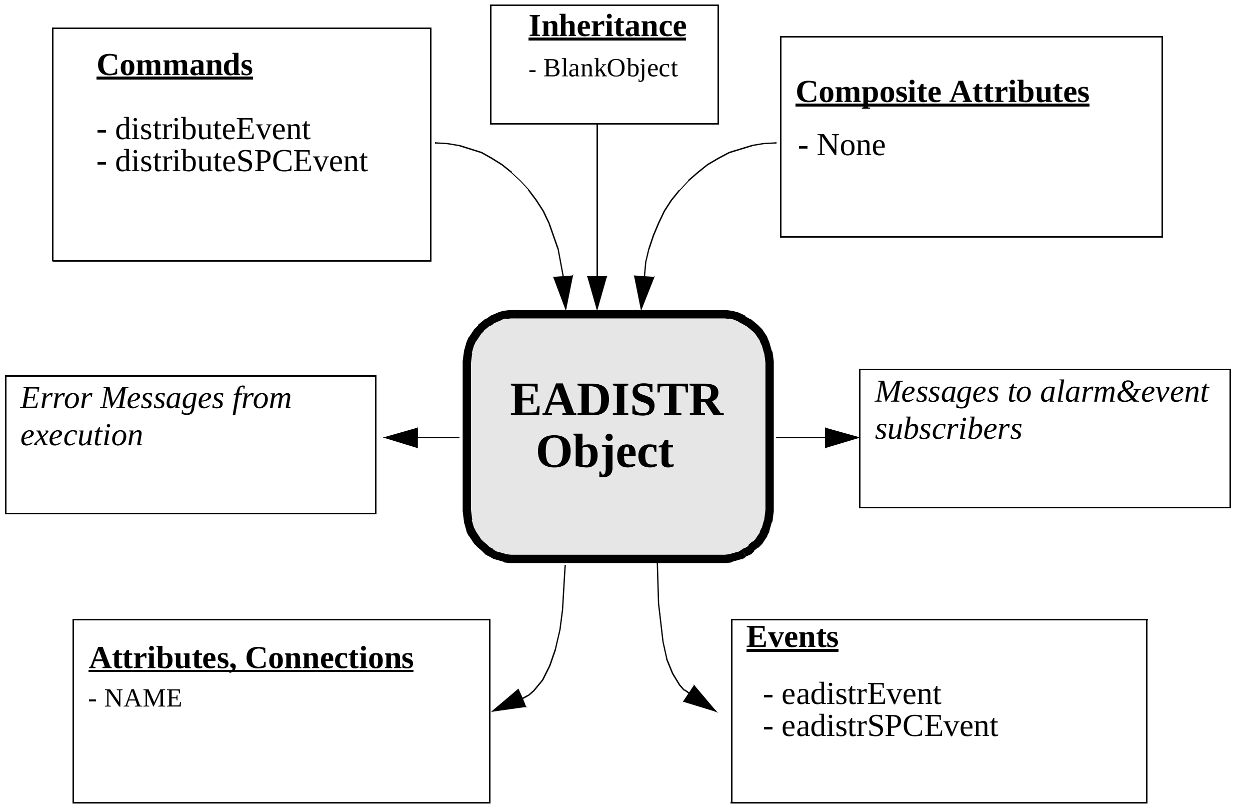


Figure 4-4. Overview of the Event and Alarm Distributor

4.3.3 Attributes

Table 4-6. EADISTR object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the EADISTR object $BCAEDISTR.

4.3.4 Connections

The EADISTR object type has no connections.

4.3.5 Composite Attributes

The EADISTR object type has no composite attributes.

4.3.6 Commands

The EADISTR object type has the commands:

distributeEvent

With **distributeEvent** you send a message to all Event and Alarm subscribers registered in the IMS node. These are normally Operator Stations in other nodes but can also be other IMS applications.

distributeSPCEvent

With **distributeSPCEvent** you send SPC messages to all Event and Alarm subscribers registered in the IMS node.

Syntax

```
distributeEvent    (objectID, objectName, objectDescription,
                   eventID, eventParameters)
```

See Table 4-4.

```
distributeSPCEvent (objectId, objectName, eventParameters)
```

See Table 4-4

Table 4-7. Parameters to the Command distributeEvent

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectID	input	TdObjectIdent - TdObjectIdent (global data type)	Object ID of object that sends the event message. The definition is found in file `/opt/advant/TypeDir/include/omfAttributeT.h` NOTE: UserAPI application should specify this as 0 (zero)
objectName	input	TdName - TdName	Name of object that sends the event message.
objectDescription	input	TdDescription - TdDescription	Description of object that sends the event message

Table 4-7. Parameters to the Command distributeEvent (Continued)

Parameter	Direction	OTDL- and C-Format	Value/explanation
eventID	input	IntShort - OmfShort	Event ID number retrieved from the type directory include files.
eventParameters	input	TdEventAlarmParType - TdEventAlarmParType (global data type)	Event parameters, see Table 4-8 below. The definition is found in file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-8. Content of the eventParameters parameter.

Parameter	OTDL- and C-Format	Value/explanation
eventReason	IntSmall - OmfSmall	Describes the kind of event that occurred. See the *AdvaBuild Object Type Builder User's Guide* for a list of possible event reasons.
eventAttribute	IntShort - OmfShort	Describes on which attribute the event occurred. See the *AdvaBuild Object Type Builder User's Guide* for a list of possible event attributes.
objectEngUnit	String[7] - OmfChar[7]	Description of the object that sends the event message.
processSection	IntSmall - OmfSmall	Process section the object belongs to.
eventValue	TdEventValueType - TdEventValueType (global data type)	Type of value that is passed. See Table 4-11.
errTreatReference	IntShort - OmfShort	Event treatment pointer in the event subscriber station. Describes how to handle the event. See Operator Station Manuals.
timeQuality	IntSmall - OmfSmall	Valid values are: 0= correct time, 1 = no time, 2 = uncertain time.
eventTime	TdBsTimeStamp - TdBsTimeStamp (global data type)	eventTime consists of two fields: - seconds since first of January 1970 - microSeconds
priority	UIntShort - UIntShort	Not used.

Table 4-8. Content of the eventParameters parameter. (Continued)

Parameter	OTDL- and C-Format	Value/explanation
compareValue	TdEventValueType - TdEventValueType (global data type)	Not used.
blockConditions	TdBlockCondition - TdBlockCondition (global data type)	Bit mask that contains 3 bits: - TdAlarmBlock states if the object's alarm handling is blocked or not. - TdPrintBlock states if the object's alarm printouts are blocked or not. - TdRepFailBlock states if repeat fail handling is blocked or not. The definition is found in file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-9. Parameters to the Command distributeSPCEvent

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectId	input	TdObjectIdent - TdObjectIdent (global data type)	ID of object that sends the SPC event message. The definition is found in file `/opt/advant/TypeDir/include/omfAttributeT.h` NOTE: UserAPI application should specify this as 0 (zero)
objectName	input	TdName - TdName	Name of object that sends the SPC event message.
eventParameters	input	TdSPCEventParType - TdSPCEventParType (global data type)	The SPC parameters, see Table 4-10 below. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-10. Content of the eventParameters parameter.

Parameter	OTDL- and C-Format	Value/explanation
eventReason	IntSmall - IntSmall	Describes the kind of SPC event that has occurred. Possible SPC event reason is: 1. Below LCL 2. Above UCL 3. Range Below LCL 4. Range Above UCL 5. Trend Down 6. Trend Up 7. Range Trend Down 8. Range Trend Up 9. Zone Low 10. Zone High 11. Range Zone Low 12. Range Zone High 13. Alarm Off 14. Change UPS status
alarmObjectId	TdObjectIdent - TdObjectIdent	ID of process object in which the event occurred. NOTE: UserAPI application should specify this as 0 (zero)
alarmObjectName	TdName - TdName	Name of process object in which the event occurred.
alarmAttributeName	TdName - TdName	Name of attribute that caused the alarm.
timeQuality	IntSmall - OmfSmall	Valid values are: 0= correct time, 1 = no time, 2 = uncertain time.
eventTime	TdBsTimeStamp - TdBsTimeStamp (global data type)	eventTime consists of two fields: - seconds since first of January 1970 - microSeconds
attributeValue	TdEventValueType - TdEventValueType (global data type)	Type of value that is passed. See Table 4-11.
limitValue	TdEventValueType - TdEventValueType (global data type)	Not used.

Table 4-11. Content of TdEventValueType

Format	Parameters	Explanation
DCX_REALV	unionTag, valueFloat	The parameters are filled in as follows: XXX.unionTag = DCX_REALV; XXX.u.valueFloat = VALUE;
DCX_INTV	unionTag, valueInteger	The parameters are filled in as follows: XXX.unionTag = DCX_INTV; XXX.u.valueInteger = VALUE;
DCX_DIGV	unionTag, valueBoolean	The parameters are filled in as follows: XXX.unionTag = DCX_DIGV; XXX.u.valueBoolean = gcTrue; (or gcFalse)

4.3.7 Events

The EADISTR has two types of events:

eadistrEvent

Raised each time the distributor receives an event message. By using the User API programming routines you can subscribe on the event.

eadistrSPCEvent

Raised each time the distributor receives an SPC event message. By using the User API programming routines you can subscribe on the SPC event.

Syntax

```
eadistrEvent  (objectID, objectName, objectDescription,
               eventID, eventParameters)
```

The parameters of the event are the same as for the command **distributeEvent** described in the Command section above.

```
eadistrSPCEvent (objectId, objectName, eventParameters)
```

The parameters of the event are the same as for the command **distributeSPCEvent** described in the Command section above.

4.3.8 EADISTR Names

One EADISTR object executes in each IMS node. It has the name: **$BCAEDISTR.** To avoid object accesses towards **$BCAEDISTR** objects in other IMS nodes all name translations for this object name should have scope set to the own node.

4.4 Calculation Object - CalcObject

4.4.1 How to Use

The Object CalcObject executes AdvaInform Calculation calculations configured with the Calculation Builder.

4.4.2 Overview

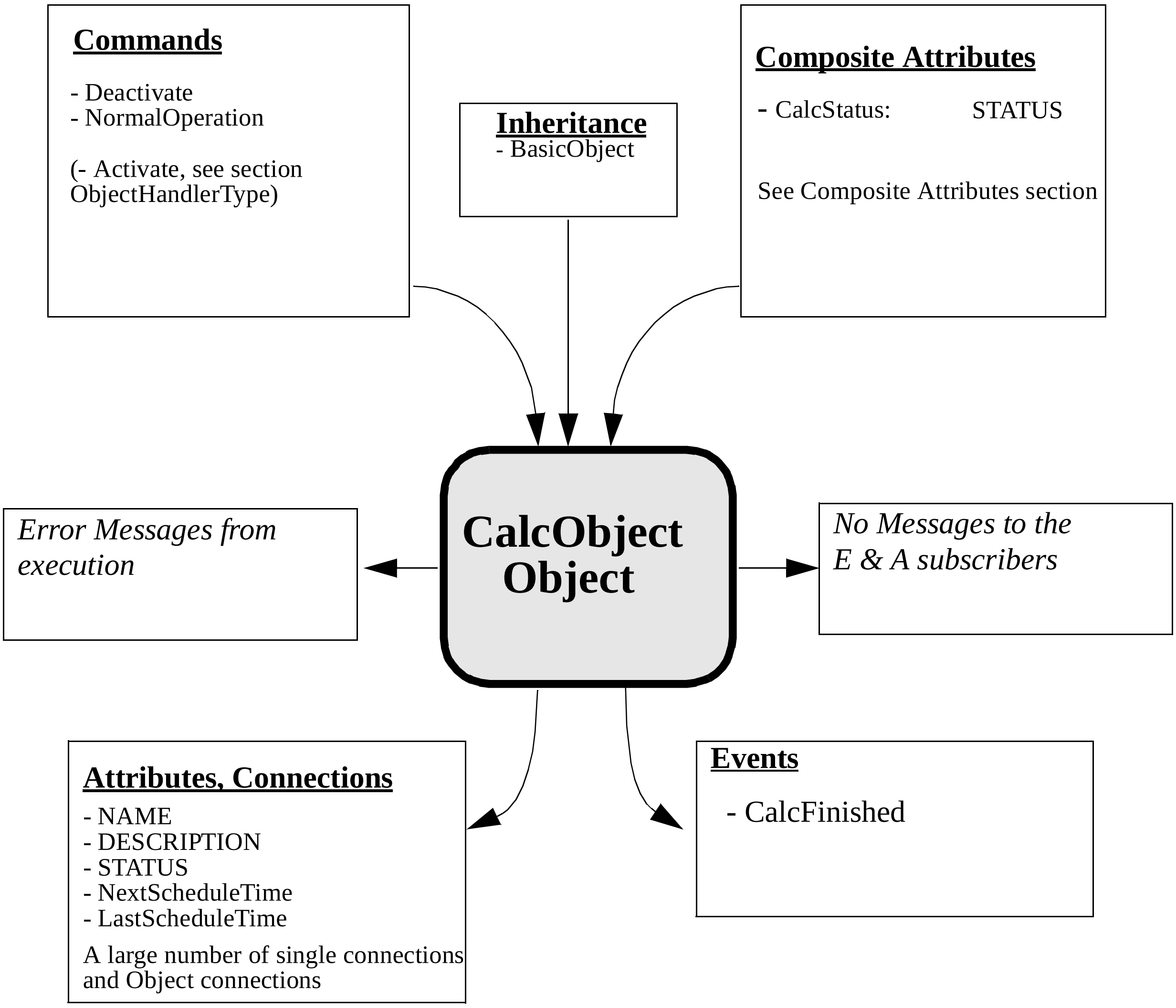


Figure 4-5. Overview of the Calculation Object

4.4.3 Attributes

Table 4-12. CalcObject object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the calculation. See the Status table below.
NextScheduleTime	String - gcString	The time for the next execution
LastScheduleTime	String - gcString	The time for the previous execution

Table 4-13. STATUS bits

Bits in TdStatus	Bit number	Description
activte	1	Object active i.e. ready to run
error	2	Indicates the overview status of the object.
updated	3	Indicates if the object has been updated since object execution started.
EXECUTION_ERROR	9	Gives the result of the last execution. True for error and False for successful execution.

4.4.4 Connections

Table 4-14. Input and output connection

Name	Direction	Type/Format	Description
InpFloat1 - 15	Input	Single/Float	Optional single input connections of float format. Used as data input if configured.
InpIntLong1 - 15	Input	Single/Float	Optional single input connections of integer long format. Used as data input if configured.
DIObj1 - 2	Input	Object/DI	Optional DI input connections used for boolean input and/or triggered activation.
AOOut1 - 2	Output	Object/AO	Optional AO output connections used for floating point output to IMS or ABB Master Process Station.
AIOut1- 2	Output	Object/AI	Optional AI output connections used for floating point output to IMS or ABB Master Process Station. The AI must be in Process Update Blocked in the PS:s.
DATOut1- 2	Output	Object/DAT	Optional DAT output connections used for integer long output to IMS or ABB Master Process Station.
DOOut1	Output	Object/DO	Optional DO output connection used for boolean output to IMS or ABB Master Process Station.
DIOut1	Output	Object/DI	Optional DI output connection used for boolean output to IMS or ABB Master Process Station. The DI must be in Process Update Blocked in the PS:s.

4.4.5 Composite Attributes

The CalcObject object type has one Composite Attribute:

CalcStatus:

STATUS

4.4.6 Commands

The CalcObject object type has commands inherited from the BasicObject. To execute a one shot execution of a calculation instance you need to execute the normal Operation of the calculation.

NormalOperation()

Deactivate()

4.4.7 Events

The CalcObject has one event: CalcFinished. It is raised each time a calculation is successfully executed.

4.4.8 CalcObject Names

Each calculation instance has a unique object name assigned during configuration.

4.5 Report Object -ReportObject

4.5.1 How to Use

The Object ReportObject executes the reports configured with AdvaInform Reports.

4.5.2 Overview

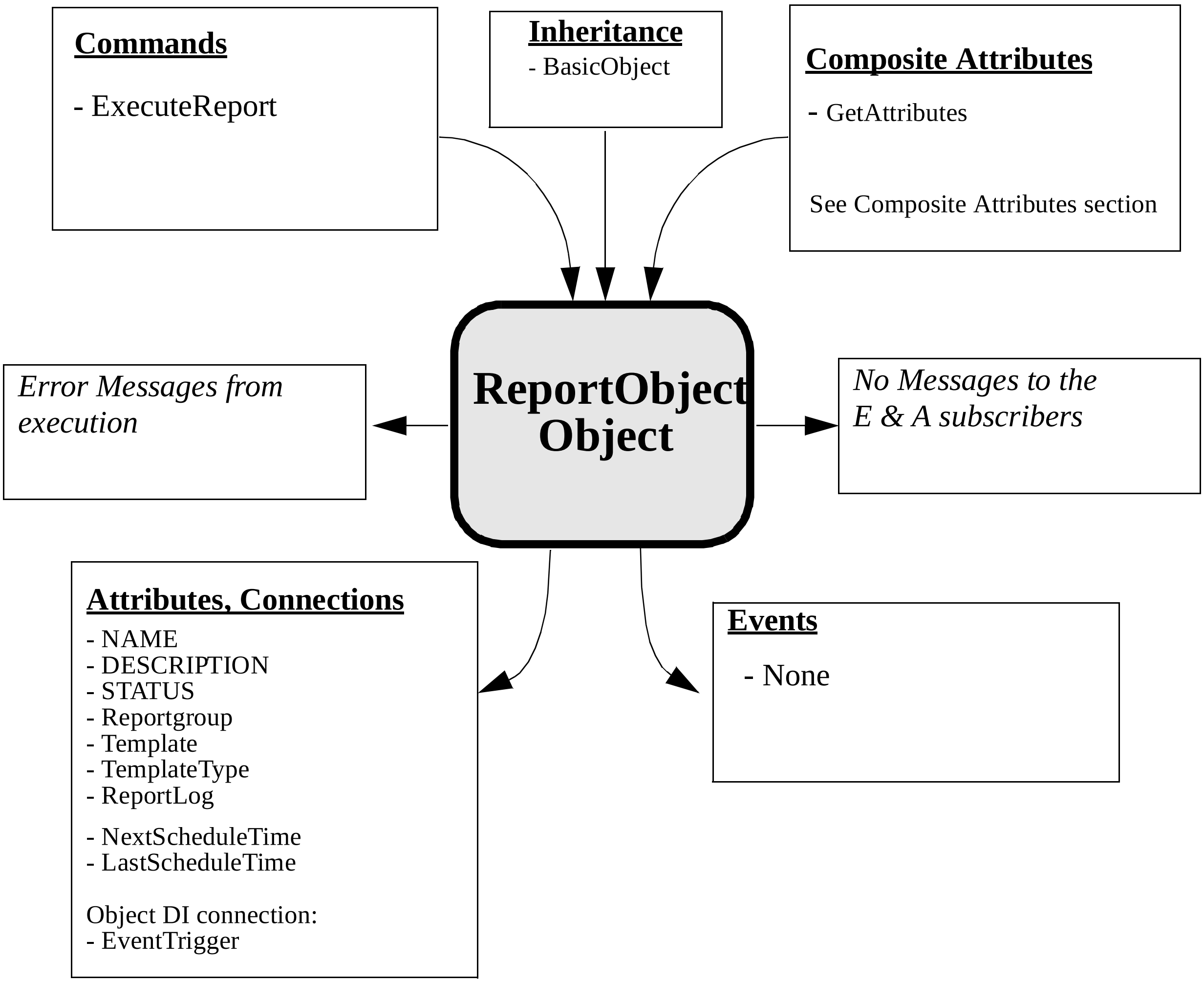


Figure 4-6. Overview of the Event and Alarm Distributor

4.5.3 Attributes

Table 4-15. ReportObject object type attributes

Attribute	OTDL Data Type	Description
NAME	TdName	The name of the object.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the Report. See the Status table below
Reportgroup	TdName	Group name for a Report object.
Template	String [81]	Template name for the Report object.
TemplateType	String [33]	Type of the template. Default is Sql*ReportWriter 1.1.
ReportLog	String [33]	Is a report log name in AdvaInform History, which report data is stored into.
ExecutionCount	IntShort	Holds the number of simultaneously executing reports.
NextScheduleTime	String[50]	The time for the next execution
LastScheduleTime	String[50]	The time for the previous execution

Table 4-16. STATUS bits

Bits in TdStatus	Bit number	Description
activte	1	Object active i.e. deactivated or activated
error	2	Indicates the overview status of the object.
updated	3	Indicates if the object has been updated since object execution started.
EXECUTION_ERROR	9	Gives the result of the last execution. True for error and False for successful execution.

4.5.4 Connections

Table 4-17. Input and output connection

Name	Direction	Type/Format	Description
EventTrigger	Input	Object/DI	Optional DI input connections used for triggered activation.

4.5.5 Composite Attributes

The ReportObject object type has one Composite Attribute:

GetAttributes:

NAME, NextScheduleTime, LastScheduleTime

4.5.6 Commands

The ReportObject object type has the following commands:

ExecuteReport

With **ExecuteReport** you request for a manual execution of a report.

Syntax

`ExecuteReport` `(PName, ParamList)`

Table 4-18. Parameters to the Command distributeEvent

Parameter	Direction	OTDL- and C-Format	Value/explanation
PName	input	TdName - TdName	The name of the Report that shall be executed.
ParamList	input	TdName - TdName	The name of the parameter List that shall be used when the report is executed.

4.5.7 Events

The ReportObject has no event.

4.5.8 ReportObject Names

Each report instance has a unique object name assigned during configuration.

4.6 Dataset Object -

4.6.1 How to Use

The Dataset objects can be configured to send or receive Datasets. The Dataset object name identifies the Dataset in the remote node, see Table 4-19.

8. Receiving: When a receiving Dataset object is configured and activated, the Dataset with that identity will be delivered to it. The received data can be stored in local DAT objects or be sent as event data to an application program.

   The receiving of Datasets can be supervised by executing the receiving object cyclically. If the received data is not updated between two cyclic executions the valid attribute is set to not valid. The cyclic execution time in the receiving Dataset should be set to 3*sendingObject cycle time.

   The Dataset object can also receive Dataset on request.

9. Sending: The sending of Datasets can either be cyclic or on request.

   a. Cyclic: If configured to send cyclic, the object sends the Dataset according to the execution interval specified with the Object Configuration tool. The connected DAT objects are used as input data.

4.6.2 Overview

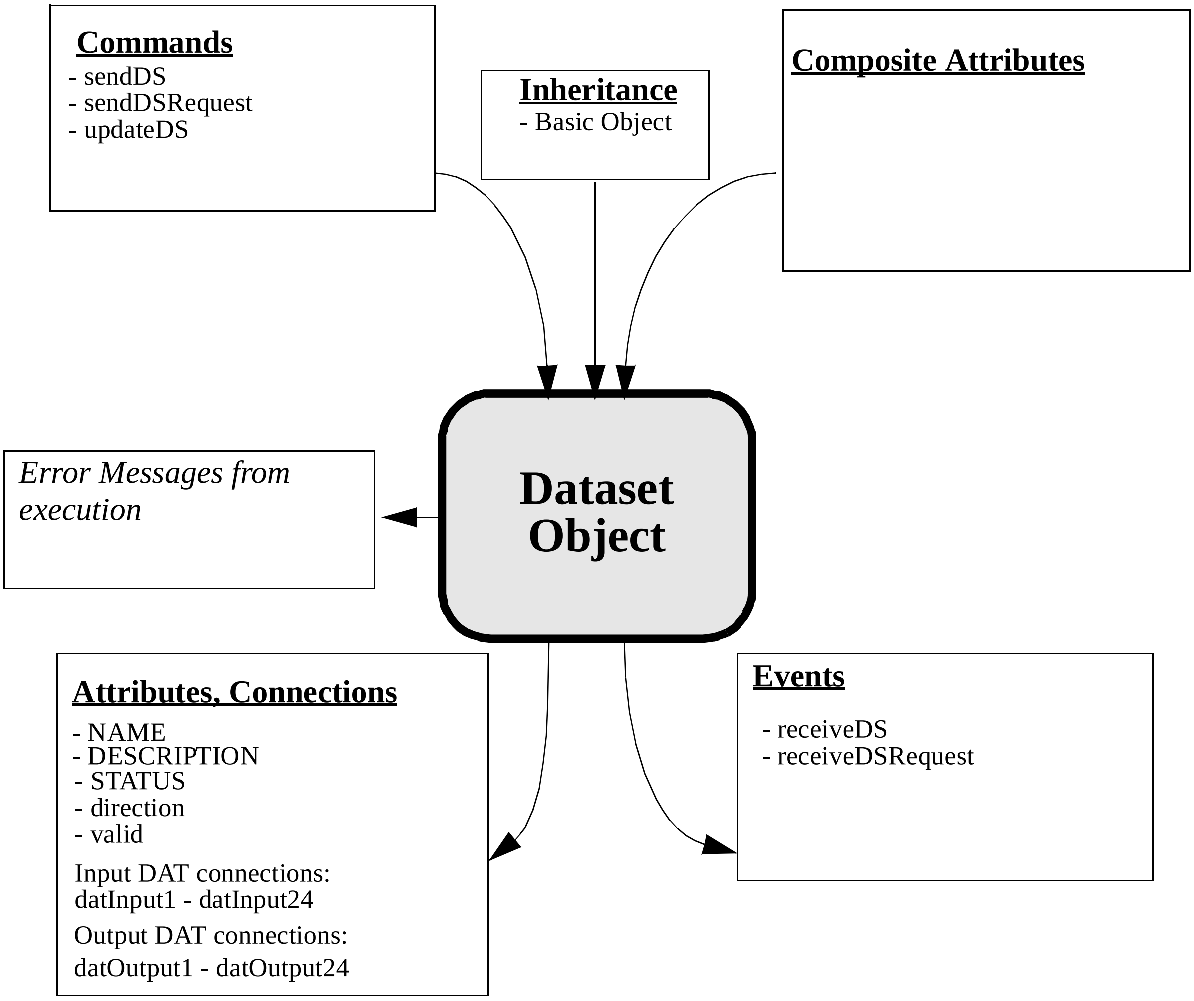


Figure 4-7. Overview of the Object Type Dataset

4.6.3 Attributes

Table 4-19. Dataset object type attributes

Attribute	OTDL Data Type	Description - Possible statuses
NAME	TdName	The name of the object. The name must follow the syntax: DATASET_*netw_node_id*. The network and node number must be between 1 and 127. The id is a sequence number for each object in a node and is between 1 and 255. The netw_node_id must match the identity of the dataset to receive.
DESCRIPTION	TdDescription	The description of the object.
STATUS	TdStatus	Represents the status of the object, see Table 4-20.
direction	IntSmall	Dataset has the following directions: DsSEND = 0 DsRECEIVE = 1
valid	IntLong	Only used if the object is configured to receive Dataset and have cyclic execution enabled. Show the validation of the received Dataset DsNOTVALID = 0 DsVALID = 1 If the Dataset is not refreshed between two cyclic executions, the valid attribute is set to not valid.

Table 4-20. TdStatus. Status bits

Bits in TdStatus	Bit number	Description
active	1	Object active i.e. ready to run.
error	2	Indicates the execution status of the object. If True, the objects execution is disturbed, indicating an internal object error.
updated	3	Indicates if the object has been updated since object execution started.

4.6.4 Connections

The Dataset has 24 optional Output DAT Connections and 24 optional Input DAT Connections.

Table 4-21. Object connection

Name	Direction	Type/Format	Description
datOutput1 - datOutput24	Output	DAT. The value from the dataset object is stored according to the DAT datatype.	Optional DAT connection. Used as output to local DAT objects if the Dataset object is configured to receive. Note! The valid flag of the received Dataset is not written to the DAT objects, only the value.
datInput1 - datInput24	Input	DAT	Optional DAT connection. Used as input from local DAT objects if the Dataset object is configured to send.

4.6.5 Operations

The operation sendDS is used to send Datasets. It can be executed from an application program. The operation has the following syntax:

```
sendDS(IntSmall numberOfDats
       IntLong dat1,
       ...
       IntLong dat24 );
```

The operation sendDSRequest has two functions depending on how the Dataset object is configured. If the object is configured to send, this operation sends the Dataset with the local DAT objects connected through datInput1 - 24 as input data.

If the object is configured to receive, the sendDSRequest operation can be used by an application program to send a request to the ABB Master Process Station for data. It has the following syntax:

```
sendDSRequest ();
```

4.6.6 Events

The Dataset object type has two events called receiveDS and receiveDSRequest. These are only raised if no input or output connections are configured for the object.

The receiveDS event is raised each time the object receives a Dataset (no output connections shall be configured). By using UserAPI routines you can subscribe on the event from an application program. The event has the following syntax:

```
receiveDS (IntSmall numberOfDats
       IntLong dat1,
       ...
       IntLong dat24 );
```

The receiveDSRequest event is raised each time the object receives a request (no input connections shall be configured). Application programs can subscribe on this event and when it is raised, the application program should send data to the dataset object via the sendDS operation. The event has the following syntax:

```
receiveDSRequest ();
```

4.6.7 User API program examples

When you install Dataset a set of AdvaInform User API program examples are installed:

- DS_sendPacket

  Sends data to a Dataset object.

- DS_getDSOnEvent

  Gets data from a Dataset object when the event receiveDS is raised.

- DS_sendPacket_OnRequest

  Sends data to a Dataset object when the event receiveDSRequest is raised.

- DS_sendReqCommand

  Sends a request for data to a number of Dataset objects.

- DS_xxxxxxx

  Reads a dataset with 24 floating point values and stores the value into 24 History logs.

4.7 System Status Object - ASNode

4.7.1 How to Use

The system status object type, ASNode, holds information about the state of an IMS node. It detects if various parts of the IMS is in good working condition or not. It allows software processes to be registered for supervision.

4.7.2 Overview

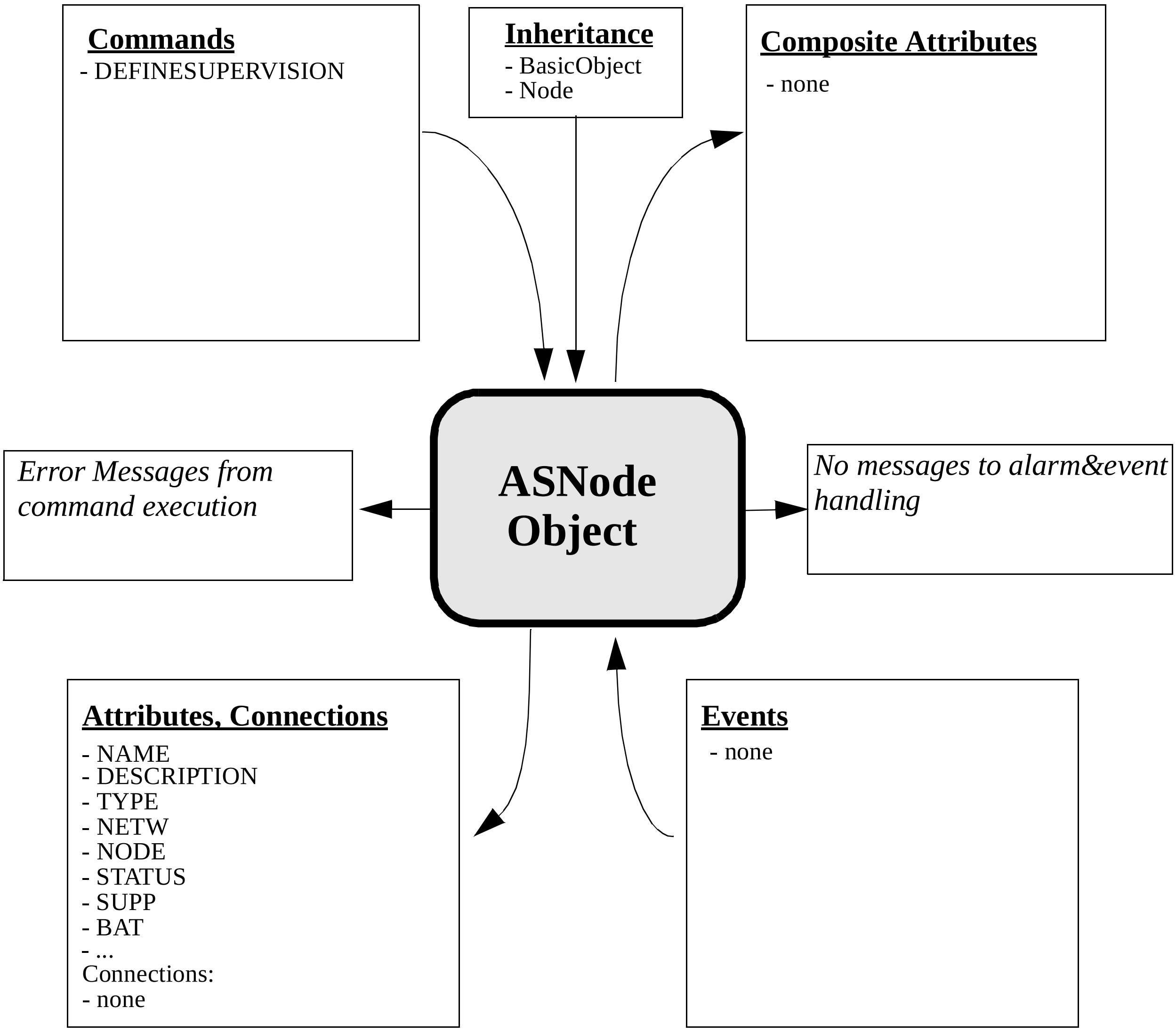


Figure 4-8. Overview of the System Status object, ASNode

4.7.3 Attributes

Table 4-22. System Status object type attributes

Attribute	OTDL Data Type	Description
NAME	OmfChar[13]	The name of the object.
DESCRIPTION	OmfChar[21]	The description of the object.
TYPE	OmfSmall	Type of node 1: MasterPiece and MasterView 2: MG 4: SuperView 8: ES 16: IMS 32: OS 64: Advant Controller
NETW	OmfSmall	Network number of node
NODE	OmfSmall	Node number of node
STATUS	NodeStatus	Status of the node. See Table 4-23
SUPP	LocDev	Power supply status
BAT	LocDev	Battery status
CPU_1	CPUDev	Status for cpu1/RTA. Status CPU 1 in HP-UX. Not implemented.
CPU_2	CPUDev	Status for cpu2/RTA. Status CPU 2 in HP-UX. Not implemented.
STATSECMEM	SummStat	Status summary for secondary memory. See Table 4-24
SECMEM	LocDevArray	Array with status for secondary memory
STATSOFTWARE	SummStat	Status summary for software
SOFTWARE	LocDevArray	Array with status for software
STATXTERM	SummStat	Summary status for x-terminals. Not implemented
XTERM	LocDevArray	Array with status for individual x-terminals. Not implemented
STATSCREEN	SummStat	Status summary for screens. Not implemented
SCREEN	LocDevArray	Array with status for individual screens. Not implemented
STATINTDEV	SummStat	Status summary for interaction devices
INTDEV	LocDevArray	Array with status for individual inter action devices
STATLOCPRT	SummStat	Status summary for local printers. Not implemented

Table 4-22. System Status object type attributes (Continued)

Attribute	OTDL Data Type	Description
LOCPRT	LocDevArray	Array with status for individual printers. Not implemented
STATPRTQUEUE	SummStat	Status summary for printer queues
PRTQUEUE	PrtQDevArray	Status for individual printer queues
REALTIMEACC	LocDev	Status for real time accelerator board
EXTUNIT	LocDev	Status of external alarms
HARDWARE	HWDev	Hardware status
STATNETWS	SummStat	Summary status for networks
NETWS	NetwArray	Network status for a maximum of 180 nodes
NETW_1	HalfNetwArray	Network status for nodes 0..89
NETW_2	HalfNetwArray	Network status for nodes 90..179
STATUS_SPARE_1	SummStat	Spare 1 summary status
SPARE_1	LocDevArray	Spare 1 individual status
STATUS_SPARE_2	SummStat	Spare 2 summary status
SPARE_2	LocDevArray	Spare 2 individual status

Table 4-23. NodeStatus bits

Bits in NodeStatus	Bit number	Description
IMPLEMENTED	1	Is set if the status is implemented
STATUS_VALID	2	Is set is the status is valid
ERROR	3	Is set for an error condition

Table 4-24. SummStat bits

Bits in SummStat	Bit number	Description
ERROR	1	Is set for an error condition
WARNING	2	Is set for an warning condition

Table 4-25. LocDevStatus bits

Bits in LocDevStatus	Bit number	Description
IMPLEMENTED	1	Is set if the status is implemented
ERROR	2	Is set for an error condition

Table 4-26. LocDev type attributes

Attribute	OTDL Data Type	Description
TYPE	IntSmall	Type of device 0, XTerm, 1, Screen, 2, OpKeyBoard, 3, UnKeyBoard, 4, AnnKeyBoard, 5, HPKeyBoard, 6, Mouse, 7, LocPrt, 8, PrtQueue, 9. PU510, 10, MP51, 11, HWDev, 12, HILDev, 13, ScsiDisk, 14, ScsiTape, 15, ScsiCDRom, 16, Scsi
STATUS	LocDevStatus	Status for device
NAME	OmfChar[21]	Name of device

Table 4-27. LocDevArray type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Maximum number of LocDev instances
last_is	OmfUShort	Number of LocDev instances
array	LocDev	Array of LocDev data entries.

Table 4-28. PrtQDev Status bits

Bits in LocDevStatus	Bit number	Description
IMPLEMENTED	1	Is set if the status is implemented
ERR_NOJOBS_EXC	2	Too much queued jobs.
ERR_MEMSIZ_EXC	3	Too much disc space used.
WARN_LESS_MEM	4	Less disc space available than defined.
WARN_NO_ENV	5	Environment variable not set or configuration variables not found. `$PRTSUPERVISION_CONFIG=/products/componentsConfig/prtSuperVision/dcPrtSupervision.cfg`
OVMS_IMPL	6	Status fetched from dxPrtQueueStatGet (i.e VAX env).
OVMS_ERROR	7	For an error condition (VAX env).

Table 4-29. PrtQDev type attributes

Attribute	OTDL Data Type	Description
TYPE	IntSmall]	
STATUS	PrtQDevStatus	Status for device
NAME	OmfChar[21]	Name of device

Table 4-30. PrtQDevArray type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Maximum number of PrtQDev instances
last_is	OmfUShort	Number of PrtQDev instances
array	LocDev	Array of LocDev data entries.

Table 4-31. CPUDevStatus bits

Bits in CPUDevStatus	Bit number	Description
CPU_IMPL	1	Set if status is implemented
CPU_PRIM	2	Set if this is the primary cpu
CPU_BACK	3	Set if this is the backup cpu
CPU_ERR	4	Set if the cpu is in an error condition
CPU_STOP	5	Set if the cpu is stopped

Table 4-32. CPUDev type attributes

Attribute	OTDL Data Type	Description
NAME	OmfChar[21]	Name of cpu device
TYPE	IntSmall	Type of cpu
STATUS	CPUDevStatus	Cpu status

Table 4-33. HWDevStatus bits

Bits in HWDevStatus	Bit number	Description
IMPLEMENTED	1	Set if the status for digital inputs from MP51 is valid
DI1	2	Digital input 1
DI2	3	Digital input 2
DI3	4	Digital input 3
DI4	5	Digital input 4
DI5	6	Digital input 5
DI6	7	Digital input 6
DI7	8	Digital input 7
DI8	9	Digital input 8
DI9	10	Digital input 9
DI10	11	Digital input 10
DI11	12	Digital input 11
DI12	13	Digital input 12
DI13	14	Digital input 13
DI14	15	Digital input 14
DI15	16	Digital input 15
DI16	17	Digital input 16
DI17	18	Digital input 17, reserved for power supply high
DI18	19	Digital input 18, reserved for battery status

Table 4-33. HWDevStatus bits (Continued)

Bits in HWDevStatus	Bit number	Description
DI19	20	Digital input 19, reserved for power supply low
DI20	21	Digital input 20, reserved for future use

Table 4-34. HWDev type attributes

Attribute	OTDL Data Type	Description
NAME_DI1	OmfChar[21]	Name of digital input 1
NAME_DI2	OmfChar[21]	Name of digital input 2
NAME_DI3	OmfChar[21]	Name of digital input 3
NAME_DI4	OmfChar[21]	Name of digital input 4
NAME_DI5	OmfChar[21]	Name of digital input 5
NAME_DI6	OmfChar[21]	Name of digital input 6
NAME_DI7	OmfChar[21]	Name of digital input 7
NAME_DI8	OmfChar[21]	Name of digital input 8
NAME_DI9	OmfChar[21]	Name of digital input 9
NAME_DI10	OmfChar[21]	Name of digital input 10
NAME_DI11	OmfChar[21]	Name of digital input 11
NAME_DI12	OmfChar[21]	Name of digital input 12
NAME_DI13	OmfChar[21]	Name of digital input 13
NAME_DI14	OmfChar[21]	Name of digital input 14
NAME_DI15	OmfChar[21]	Name of digital input 15
NAME_DI16	OmfChar[21]	Name of digital input 16
NAME_DI17	OmfChar[21]	Name of digital input 17
NAME_DI18	OmfChar[21]	Name of digital input 18
NAME_DI19	OmfChar[21]	Name of digital input 19
NAME_DI20	OmfChar[21]	Name of digital input 20
STATUS	HWDevStatus	Status for digital inputs from MP51

Table 4-35. NetwStatus bits

Bits in NetwStatus	Bit number	Description
IMPLEMENTED	1	Set if status is valid
ERROR	2	Set if there is a network error
BUS_A_ERR	3	Set if there is an error in bus a. Valid only for redundant MB200.
BUS_B_ERR	4	Set if there is an error in bus b. Valid only for redundant MB200.
MASTER	5	Set if the node is busmaster. Valid only for MB200.
MB100	6	Set if the bus is MB100
MB200	7	Set if the bus is MB200
MB300	8	Set if the bus is MB300
MNET	10	Set for old MasterNet equipment e.g ADC
UNIDENTIFIED	11	Set if the bus is unidentified
GCOM	12	Set if the bus is GCOM
RCOM	13	Set if the bus is RCOM
MVI	14	Set if the bus is MVI

Table 4-36. Netwtype attributes

Attribute	OTDL Data Type	Description
TYPE	IntSmall	Type of bus -1, Unknown 0, MNET 1, MB100 2, MB200 3, MB300 6, GCOM 7, RCOM 8, MVI
NETW	IntSmall	Network
NODE	IntSmall	Node
STATUS	NetwStatus	Status for node

4.7.4 Connections

The system status object has no connections.

4.7.5 Composite Attributes

The system status object has no Composite Attributes.

4.7.6 Commands

The system status object has the following command.

DEFINESUPERVISION

With **DEFINESUPERVISION** it is possible to define software processes to be supervised by the system status object. It is possible to give a number of processes a common group name. If one of the processes in the group is not executing the group will receive a negative status.

Syntax

```
DEFINESUPERVISION (typestr, group_name, name, action)
```

Table 4-37. Parameters to the Command DEFINESUPERVISION

Parameter	Direction	OTDL- and C-Format	Value/explanation
typestr	input	String[20] - OmfChar[20]	Should always have the value “SOFTWARE”.
group_name	input	String[80] - OmfChar[80]	The name of a group of software processes. If the software process to be registered is not a member of a group the group name parameter should be a empty string.
name		String[80] - OmfChar[80]	The name of the software process to be supervised.
action		SupervisionAction - OmfNodeSupervisionAction	Specifies if a process name should be added to, **addName**, or removed from, **removeName**, the supervised processes.

4.7.7 Events

The System Status object has no events.

4.7.8 System Status Names

One System Status object executes in each IMS node. It has the name **NODE_nn** where nn is the number of the node where the System Status object resides**.**

4.8 Log Manager Object - LogMgr

4.8.1 How to Use

The Log Manager is the main retrieval interface object towards AdvaInform History. Through the Log Manager, History Numeric and Message data can be retrieved to a UserAPI user application. To update Numeric or Message data from an application the Numeric Log or Message Log objects can be used directly but also via UserAPI.
One Log Manager exists in each IMS node.

4.8.2 Overview

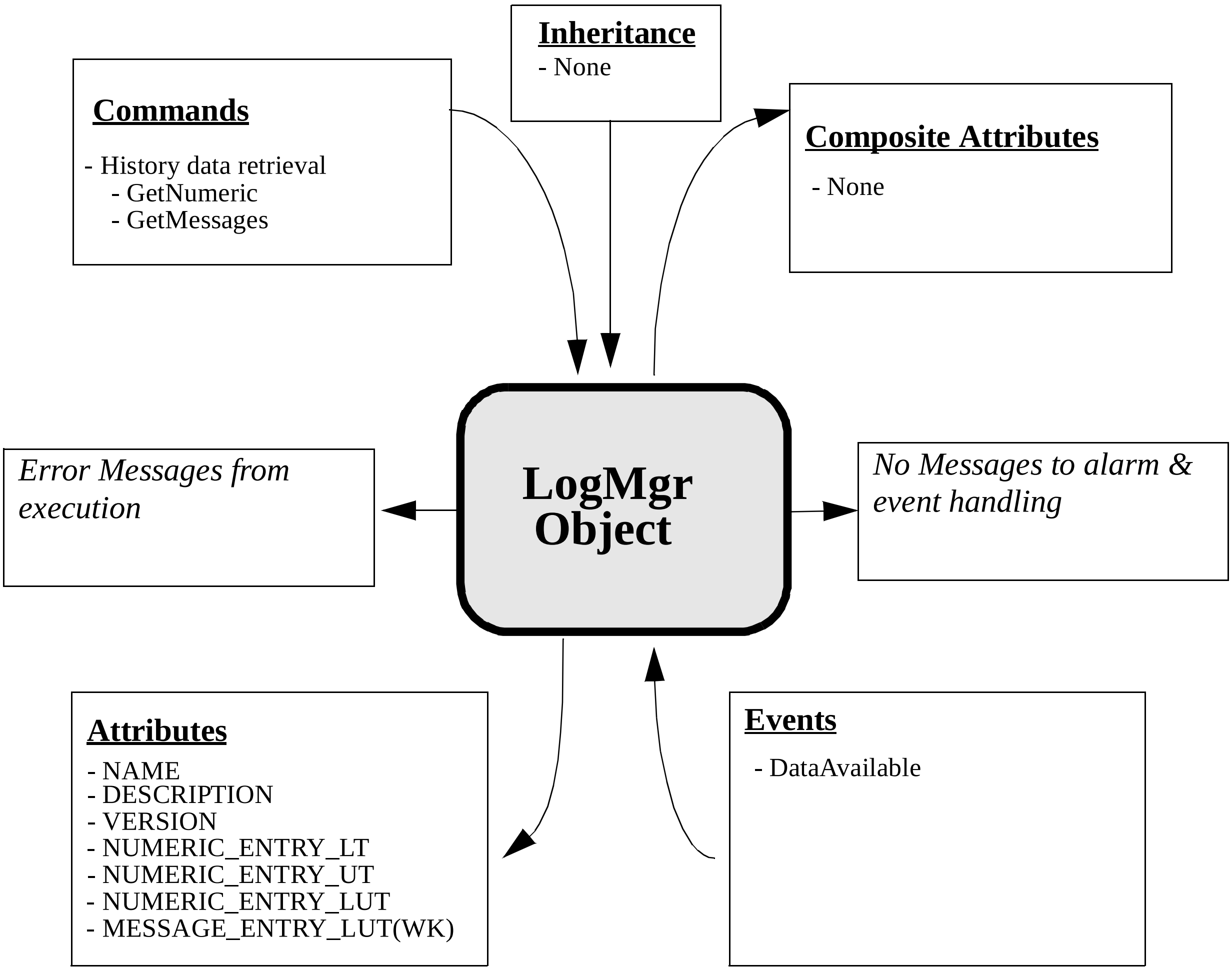


Figure 4-9. Overview of the Log Manager, LogMgr

4.8.3 Attributes

Table 4-38. Log Manager object type attributes

Attribute	OTDL - C Data Type	Description
NAME	OmfChar(33)	Name of Log Manager. NOTE: name is always ‘LOG_MANAGER’.
DESCRIPTION	OmfChar(41)	Description of Log Manager.
NUMERIC_ ENTRY_LT	HsNumeric EntryLT	Open array of max 500 historical values with Local timestamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
NUMERIC_ ENTRY_UT	HsNumeric EntryUT	Open array of max 500 historical values with UTC timestamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
NUMERIC_ ENTRY_LUT	HsNumeric EntryLUT	Open array of max 500 historical values with Local & UTC timestamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
MESSAGE_ ENTRY_LUT	HsMessage EntryLUT	Open array of max 225 historical messages with Local & UTC timestamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
MESSAGE_ ENTRY_LUTWK	HsMessage EntryLUTWk	Open array of max 225 historical messages with Local & UTC timestamp and all different parts of the messages data as separate attributes. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-39. Log Manager HsNumericEntryLT / UT / LUT Open Array type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Max index (dimension - 1) of the array holding data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 499.

Table 4-39. Log Manager HsNumericEntryLT / UT / LUT Open Array type attributes (Continued)

Attribute	OTDL - C Data Type	Description
last_is	OmfUShort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	HsNumericLT OR HsNumericUT OR HsNumericLUT	Array of History numeric data entries.

Table 4-40. Log Manager HsMessageEntryLUT / HsMessageLUTWk
Open Array type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Max index (dimension - 1) of the array holding data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 224 for LUT and 124 for LUTWk.
last_is	OmfUShort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	HsMessageLUT OR HsMessageLUTWk	Array of History message data entries.

Table 4-41. Log Manager HsNumericLT / UT / LUT type attribute

Attribute	OTDL - C Data Type	Description
localTime	HsTimeStamp - BciTimeStamp	Timestamp of this numeric value entry in Local time format. The localTime attribute exists in both HsNumericLT and HsNumericLUT NOTE: The localTime attribute is not present in HsNumericUT.
utc	HsTimeStamp - BciTimeStamp	Timestamp of this numeric value entry in UTC time format. The utc attribute exist in both HsNumericUT and HsNumericLUT. NOTE: The utc attribute is not present in HsNumericLT.

Table 4-41. Log Manager HsNumericLT / UT / LUT type attribute (Continued)

Attribute	OTDL - C Data Type	Description
entryValue	OmfFloat	Numeric data value entry. NOTE: A zero (0.0) data value may also be a noData or badData, the entryStatus attribute indicates this.
entryStatus	HsLogStatus Type	Status of this numeric data value entry. For a description of the HsLogStatusType type please refer to the HsLogStatusType bits table in the Numeric Log section. See Table 4-48
objectStatus	HsObjectStatus Type	The status of the data source object. Not valid when data source is a TTD log. HsObjectStatusType bits table in the Numeric Log section. See Table 4-49

Table 4-42. Log Manager HsMessageLUT / HsMessageLUTWk type attribute

Attribute	OTDL - C Data Type	Description
localTime	HsTimeStamp - BciTimeStamp	Timestamp of this message value entry in Local time format.
utc	HsTimeStamp - BciTimeStamp	Timestamp of this message value entry in UTC time format.
historyMsgID	OmfULong	Identifier of this message. Used for further references.
msgCount	OmfUSmall	Not used.
entryStatus	HsLogStatus Type	Status of this message data entry. For a description of the HsLogStatusType type please refer to the HsLogStatusType bits table in the Numeric Log section. See Table 4-48
message	OmfChar(255)	Message string data entry.
procObjName	OmfChar(33)	Name of process object generating the message. NOTE: the procObjName is only present in HsMessageLUTWk
sectionName	OmfChar(33)	The section the process object belongs to. NOTE: the sectionName is only present in HsMessageLUTWk
batchID	OmfChar(41)	The batch identity. Not used. NOTE: the batchID is only present in HsMessageLUTWk

Table 4-42. Log Manager HsMessageLUT / HsMessageLUTWk type attribute (Continued)

Attribute	**OTDL - C Data Type**	**Description**
unitID	OmfChar(13)	The unit identity Not used. NOTE: the unitID is only present in HsMessageLUTWk
sequenceID	OmfChar(13)	The sequence identity. Not used. NOTE: the sequenceID is only present in HsMessageLUTWk
eventReason	OmfULong	The reason for the event generating the message. NOTE: the eventReason is only present in HsMessageLUTWk
eventProperty	OmfULong	The property involved in the event generating the message. NOTE: the eventProperty is only present in HsMessageLUTWk
netwID	OmfULong	The identity of the network where the message was generated. NOTE: the netwID is only present in HsMessageLUTWk
nodeID	OmfULong	The identity of the node where the message was generated. NOTE: the nodeID is only present in HsMessageLUTWk
priority	OmfULong	The priority of the message. NOTE: the priority is only present in HsMessageLUTWk
msgType	HsMessageType	The type of the message. hsALARM = 1, hsBILLBOARD = 2, hsPARAMETER_CHANGE = 3, hsLADDER_LOGIC = 4, hsSPC = 5, hsALARM_ACK = 6, hsDIAGNOSTIC = 7, hsBATCH_MSG = 8. NOTE: the msgType is only present in HsMessageLUTWk

4.8.4 Commands

The Log Manager has the following commands:

GetNumeric

With **GetNumeric** it is possible to retrieve up to 500 historical numerical data values from a History Numeric Log. The request command must be combined with an on event subscription for one of the attributes **NUMERIC_ENTRY_LT, NUMERIC_ENTRY_UT** or **NUMERIC_ENTRY_LUT** with the event **DataAvailable**.

Syntax

```
GetNumeric  ( ObjectName.objectName, SelectionTime,
            TimeJustification, TimeType,
            TimeInterv, CyclicTimeInterv,
            InitialValueEntries, InitalEventEntries,
            CyclicValueEntries, CyclicEventEntries,
            RetrievalType, InterpolationAlg,
            LogEntryOrder, DirectionOfRequest,
            dataSelector, queryStart (calcAlg), AccessID )
```

See Table 4-4

GetMessages

With **GetMessages** it is possible to retrieve up to 225 historical messages from a History Message Log. The request command must be combined with an on event subscription for one of the attributes **MESSAGE_ENTRY_LUT** or **MESSAGE_ENTRY_LUTWK** with the event **DataAvailable**.

Syntax

```
GetMessages ( objectName, selectionTime,
            timeType, timeJustification,
            timeInterv, historyMsgID,
            messageType, retrieveKeyType,
            rettrieveKey, initialMsgEntries,
            initalEventEntries, cyclicMsgEntries,
            cyclicEventEntries, logEntryOrder
            directionOfRequest, queryStart, language )
```

NOTE

To retrieve History numeric or message data from the Log Manager using a UserAPI application, the **bciGetAttribOnEventWithInit()** UserAPI function should be used with the following parameters/data:

- Either of the attributes **NUMERIC_ENTRY_XXT** or **MESSAGE_ENTRY_XXX**.
- The init command/operation **GetNumeric** or **GetMessages**.
- The event **DataAvailable**.
- Include files **omfTypeDirectory.h** and **omf_LogMgr.h**.

For a description of the GetNumeric and GetMessages parameters please refer to the sections covering the **Numeric Log** and **Message Log** object commands.

Table 4-43. Parameters to the Command GetNumeric

Parameter	Direction	OTDL- and C-Format	Value/explanation
ObjectName.objectName	input	HsTimeStamp - HsLogNameType	This string contains a name which directs the getNumeric request to the appropriate numeric or numeric type log (like TTD). For example, today, this name takes one of the following types of values: TTD name (TTD_2_2_2_2): The logManager forwards the getNumeric to the ttdManager. History NumericLog or Restored NumericLog Name ($HSTAG1, VALUE-1-r, $HSAIC21_1, VALUE-3-0): The logManager forwards the getNumeric operation to the historyServer. AccessName (AIC21_1, VALUE): The logManager attempt to connect any TTD and History Logs with the same accesName. The attributes of each of the logs are fetched (storageInterval, calcAlg, logPeriod etc.) and the “SEAMLESS” algorithm is applied to chose a log to meet the criteria of the getNuemricRequest.
SelectionTime	input	HsTimeStamp - HsTimeStamp	History stored data in local and UTC times. If the timeType is hsPRESENT, the selectionTime can be viewed as NOW in UTC time. When the retrieval process receives PRESENT, it get the current time at that time. If the timeType is LOCAL, the Time is interpreted as a localTime. If the timeType is UTC, the selectionTime is interpreted as a UTC time.
TimeType	input	HsTimeType - HsTimeType	
TimeJustification	input	HsTImeJust - HsTimeJust	These two parameters, combined with the TimeType and SelectionTime set up the START and END time for the data requested. If the timeType is START_TIME, the selectionTime is the START_TIME and the END_TIME is set START_TIME + TimeInterv. If the timeType is END_TIME, the END_TIME is the selectionTime and the START_TIME is the END_TIME - TimeInterv. If timeInterv == 0, timeInterv is interpreted as infinit.
TimeInterv	bidirect	HsTimeInterval - HsTimeIterval	

Table 4-43. Parameters to the Command GetNumeric (Continued)

Parameter	Direction	OTDL- and C-Format	Value/explanation
CyclicTimeInterv	bidirect	HsTimeInterval - HsTimeInterval	The cyclicTimeInterval moves the select window FORWARD/BACKWARD (dirOfRequest) from the initial startTime/endTime setup above. A cyclic FORWARD request will return data from END_TIME (above) to END_TIME + cyclicTimeInterv for the first cyclicInterval and END_TIME + cyclicTimeInterv to END_TIME + 2 x cyclicTimeInterv, etc.
DirectionOfRequest	input	HsRequestDir - HsRequestDir	hsFORWARD hsBACKWARD Direction to continue with cyclic return of data. Forward or backward.
InitialValueEntries	input	UIntShort - OmfUShort	No of values retrieval at first update (max 500).
InitalEventEntries	input	UIntShort - OmfUShort	Not used = 0
CyclicValueEntries	input	UIntShort - OmfUShort	No of values retrieval at each cyclic update.
CyclicEventEntries	input	UIntShort - OmfUShort	Not used = 0

Table 4-43. Parameters to the Command GetNumeric (Continued)

Parameter	Direction	OTDL- and C-Format	Value/explanation
RetrievalType	input	HsRetrievalType - HsRetrievalType	The user is capable of using three types of retrievalTypes: RAW, INTERPOLATED and DISPLAY. Definitions: **RAW**: What you see is what you get. Any data between the StartTime and EndTime you specify is returned to you. If there is no data in the interval, you get nothing. **INTERPOLATED**: (interpolation algorithm) The interpolated retrieval type exists to reduce the number of points returned to the display element. For example, Two logs exist, a 1 second 1 hour log and a 15 minute 1 day log. The display can only receive up to 500 numeric points at a time. The user wants 30 minuter of data from the last half hour. If a RAW request was used, the 15m log would return two points and the 1s log would return 1800 points. 2 points fit in the display, but no curve detail is present. 1800 points do not fit in the display and only 5/18 of the data is available.

Table 4-43. Parameters to the Command GetNumeric (Continued)

Parameter	Direction	OTDL- and C-Format	Value/explanation
			An interpolated/AVG request on the 1s second log would solve this problem by doing a time average of the 1800 points and reducing the number of points to 500 (or initialValueEntries) if the intialValue entries was 30, blocks of 60 points would be averaged. This would reduce the points returned from 1800 to 30. **DISPLAY**: The display retrieval type gives the logManager the power to decide whether to use RAW or INTERPLATED/AVG to retrieve data from a given Numeric log. If the objectName is a logName, display is changed to RAW if the stored points in the interval are less than or equal to the points requested. If the points in the interval are greater than the points requested, the retrieval type is changed to INTERPOLATED/AVG before the request is forwarded to the log. If the objectName is an accessName, the logManager gathers a list of all TTD and history logs and sorts them in the order best suited to fill the request. By process of elimination, the log Manger finds the best log to fill the request. Depending on the storage interval of the log which was chosen to receive the request, the retrieval type is change to RAW or INTERPALTED/AVG for the same reasons as the logName case. An additional feature of the DISPLAY retrieval type is for PRESENT/END_TIME requests, TTD data will be used or the gap between the last history point returned and there present time and to supply cyclic updates. If no history logs exist for an accessName, but TTD does, the logManager will use TTD to give the user what it can.
InterpolationAlg	input	HsInterpAlg - HsInterpAlg	hsAVERAGE = Average hsMAXIMUM = Maximum hsMINIMUM = Minimum hsSum = SUM hsSUM_OF_SQRS = Sum of SQARE Roots hsSTANDARD_DEV = Standard deviation hsNUM_OF_VALUES = Number of values hsWILDCAR = Wildcard

Table 4-43. Parameters to the Command GetNumeric (Continued)

Parameter	Direction	OTDL- and C-Format	Value/explanation
LogEntryOrder	input	HsEntryOrder - HsEntryOrder	hsASCENDING = Sort data in ascending order hsDESCENDING = Sort
dataSelector	input	HsDataSelector - HsDataSelector	When retrieval selects data from the database, ORIGINAL will retrieve only the original data from the database. (NO MODIFIED VALUES). CURRENT will take the current value for each time in the database.
queryStart (calcAlg)	input	HsTimeJust - HsTimeJust	queryStart has been overloaded. Its definition has been changed to be calculation algorithm. A calculation must be specified when retrieving by accessName to choose a MIN log over a AVG log or a SUM over a MAX log.
AccessID	output	UIntShort - OmfUShort	

4.8.5 Events

Table 4-44. The event DataAvailable

Name	OTDL - C Data type	Value/explanation
DataAvailable	HsTimeStamp - BciTimeStamp	The Numeric or Message data that was requested with command GetNumeric or GetMessages is available in the attributes NUMERIC_ENTRY_LT / UT / LUT or MESSAGE_ENTRY_LUT / LUTWk. The event itself contains a time stamp.

4.8.6 Log Manager Names

One Log Manager object executes in each IMS node.
It has the name: **LOG_MANAGER.**

When accessing a LOG_MANAGER in a specific node, use the “scope”, i.e. network and node number.

4.9 Numeric Log Object - NLog

4.9.1 How to Use

Each Numeric Log object defines one numerical log within AdvaInform History. Through the Numeric Log, History numeric data can be updated from a UserAPI application. One Numeric Log object exists for each numeric log defined in the IMS station.

Numeric Logs are created and named through AdvaInform History builder functions.

The AdvaInform SQL*Connect - Client makes AdvaInform History Numeric Logs appear as a relational database with read access. The database table name used for Numeric Logs is NUMLOGVAL. The columns ret_type, interp_alg, interval, selection and calc_alg are used for defining parameters for fetching data.

4.9.2 Overview

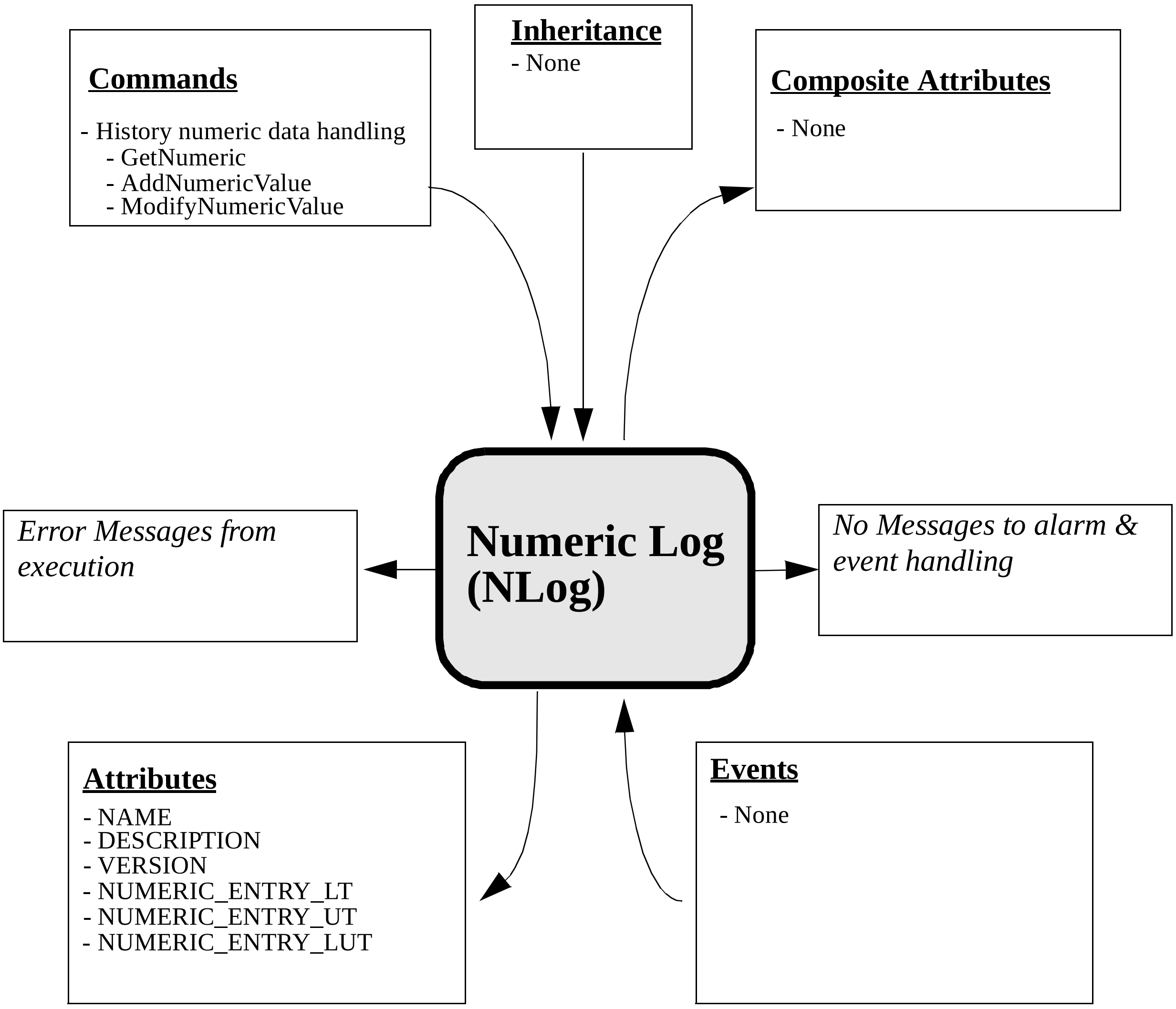


Figure 4-10. Overview of the Numeric Log

4.9.3 Attributes

Table 4-45. Numeric Log object type attributes

Attribute	OTDL - C Data Type	Description
NAME	OmfChar(81)	Name of Numeric Log.
DESCRIPTION	OmfChar(41)	Description of Numeric Log.
NUMERIC_ ENTRY_LT	HsNumeric EntryLT	Open array of max 500 historical numeric values with Local time stamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
NUMERIC_ ENTRY_UT	HsNumeric EntryUT	Open array of max 500 historical numeric values with UTC time stamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
NUMERIC_ ENTRY_LUT	HsNumeric EntryLUT	Open array of max 500 historical numeric values with Local & UTC time stamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-46. Numeric Log HsNumericEntryLT / UT / LUT Open Array type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Max index (dimension - 1) of the array holding data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 499.
last_is	OmfUShort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	HsNumericLT OR HsNumericUT OR HsNumericLUT	Array of History numeric data entries.

Table 4-47. Numeric Log HsNumericLT / UT / LUT type attribute

Attribute	OTDL - C Data Type	Description
localTime	HsTimeStamp - BciTimeStamp	Time stamp of this numeric value entry in Local time format. The localTime attribute exists in both HsNumericLT and HsNumericLUT. NOTE: The localTime attribute is not present in HsNumericUT.
utc	HsTimeStamp - BciTimeStamp	Time stamp of this numeric value entry in UTC time format. The utc attribute exist in both HsNumericUT and HsNumericLUT. NOTE: The utc attribute is not present in HsNumericLT.
entryValue	OmfFloat	This numeric data value entry. NOTE: A zero (0.0) data value may also be a noData or badData, the entryStatus attribute indicates this.
entryStatus	HsLogStatus Type	Status of this numeric data value entry. Value is a bitmap as described in the following table
objectStatus	HsObjectStatus Type	The status of the data source object. Not valid when data source is a TTD log.

Table 4-48. HsLogStatusType bits

Bits in HsLogStatusType	Bit number	Description
noData	1	No valid data.
badData	2	Bad data in this entry.
entryArchived	3	Entry is archived.
valueInAlarm	4	Value is in alarm.
alarmPriority0	5	Alarm of priority 0.
alarmPriority1	6	Alarm of priority 1.
timeQuality	7	Uncertain time quality.
timeChange	8	A time change has occurred.
sourceTimeStamp	9	Entry was time stamped at source.
timeStampAdded	10	A time stamp has been added.
_dummy0	11 - 12	Not used.
userEntered	13	Data was user entered.
modifiedData	14	Data has been modified.

Table 4-48. HsLogStatusType bits (Continued)

Bits in HsLogStatusType	Bit number	Description
modifiedAvailable	15	Modified data exists.
_dummy1	16	Not used.
firstEntry	17	This is the first entry.
lastEntry	18	This is the last entry.
valueInterpolated	19	Data value has been interpolated.
_dummy2	20	Not used.
sequenceNum0	21	Sequence number bit 0, value 1.
sequenceNum1	22	Sequence number bit 1, value 2.
sequenceNum2	23	Sequence number bit 2, value 4.
sequenceNum3	24	Sequence number bit 3, value 8.
sequenceNum4	25	Sequence number bit 4, value 16.
TTDValue	26	Data value origins from Master TTD.
CCFValue	27	Data value origins from MOD CCF.
_dummy3	28- 32	Not used.

Table 4-49. HsObjectStatusType bits

Bits in HsLogStatusType	Bit number	Description
_dummy 0	1 - 32	Describes the status of the data source object. See the description of the status bits for the data source object type.

4.9.4 Commands

The Numeric Log has the following commands:

GetNumeric

With **GetNumeric** it is possible to retrieve up to 500 historical numerical data values from a History Numeric Log. The request command must be combined with an on event subscription for one of the attributes **NUMERIC_ENTRY_LT, NUMERIC_ENTRY_UT** or **NUMERIC_ENTRY_LUT** with the event **DataAvailable**.

Syntax

```
GetNumeric  ( objectName, selectionTime,
            timeJustification, timeType, timeInterv,
            cyclicTimeInterv, initialValueEntries,
            initalEventEntries, cyclicValueEntries,
            cyclicEventEntries, retrievalType,
            interpolationAlg, logEntryOrder,
            directionOfRequest, dataSelector, queryStart
            (calcAlg), accessID )
```

AddNumericValue

With **AddNumericValue** it is possible to update up to 500 historical numerical data values into a History Numeric Log.

Syntax

```
AddNumericValue ( objecName, handle, timeType, logEntryArray )
```

ModifyNumericValue

With **ModifyNumericValue** it is possible to modify up to 500 historical numerical data values in a History Numeric Log. The Value and the Object Status are possible to change. The Entry Status is included in the numerical data structure but has no effect on the numerical log. See the note below for more information.

Syntax

```
ModifyNumericValue (objectName, handle, timeType, logEntryArray)
```

NOTE

To update History numeric or message data from the Log Manager using a UserAPI application, the **bciGetAttribOnEventWithInit()** UserAPI function should be used with the following parameters/data:

- - Either of the attributes **NUMERIC_ENTRY_XXT** or **MESSAGE_ENTRY_XXX**.
- The init command/operation **GetNumeric** or **Get Messages**.
- The event **DataAvailable**
- Include files **omfTypeDirectory.h** and **omf_LogMgr.h**.

For a description of the GetNumeric and GetMessages parameters please refer to the sections covering the **Numeric Log** and **Message Log** object commands..

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric

Parameter	Direction	OTDL- and C-Format	Value/explanation

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric (Continued)

objectName.objectName	input	HsTimeStamp - HsLogNameType	This string contains a name which directs the getNumeric request to the appropriate numeric or numeric type log (like TTD). For example, today, this name takes one of the following types of values: TTD name (TTD_2_2_2_2): The logManager forwards the getNumeric to the ttdManager. History NumericLog or Restored NumericLog Name ($HSTAGI1, VALUE-1-r, $HSAIC21_1, VALUE-3-0): The logManager forwards the getNumeric operation to the historyServer. AccessName (AIC21_1, VALUE): The logManager attempt to connect any TTD and History Logs with the same accessName. The attributes of each of the logs are fetched (storageIntrval, calcAlg, logPeriod etc.) and the “SEAMLESS” algorithm is applied to choose a log to meet the criteria of the getNumericRequest.
selectionTime	input	HsTimeStamp - HsTimeStamp	History stored data in local and UTC times.If the timeType is hsPRESENT, the selectionTime can be viewed as NOW in UTC time. When the retrieval process receives PRESENT, it get the current time at that time. If the timeType is LOCAL, the time is interpreted as a localTime. If the timeType is UTC, the selectionTime is interpreted as a UTC time.
timeType	input	HsTimeType - HsTimeType	

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric (Continued)

timeJustification timeInterv	input Input	HsTimeJust - HsTimeJust HsTimeInterval - HsTimeInterval	These two parameters, combined with the timeType and selectionTime set up the START and END time for the data requested. If the timeType is START_TIME, the SelectionTime is START_TIME and the END_TIME is set START_TIME + timeInterv. If the timeType is END_TIME, the END_TIME is the selectionTime and the START_TIME is the END_Time - timeInterv. IF timeInterv = O, TimeInterv is interpreted as infinit.
cyclicTimeInterv directionOfRequest	bidirect input	HsTimeInterval - HsTimeInterval HsRequestDir - HsRequestDir	The cyclicTimeInterval moves the select window FORWARD/BACKWARD (directionOfRequest) from the initial startTime/endTime setup above. A cyclic FORWARD request will return data from END_TIME (above) to END_TIME + cyclicTimeInterv for the first cyclicInterval and END_TIME + cyclicTimeInterval to END_TIME + 2 x cyclicTimeInterval, etc. hsFORWARD hsBACKWARD Direction to continue with cyclic return of data Forward and Backward.
initialValueEntries	input	UIntShort OmfUShort	Number of values retrieval at first update. NOTE: maximum is 500.
initialEventEntries	input	UIntShort - OmfUShort	Not used = 0
cyclicValueEntries	input	UInt Short - OmfUShort	Number of values retrieval at each cyclic update.
cyclicEventEntries	input	UIntShort - OmfUShort	Not used = 0

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric (Continued)

retrievalType	input	HsRetrievalType - HsRetrievalType	The user is capable of using three types of retrievalTypes: RAW, INTERPOLATED and DISPLAY. Definitions: **RAW**: What you see is what you get. Any data between the StartTime and EndTime you specify is returned to you. If there is no data in the interval, you get nothing. **INTERPOLATED**: (interpolation algorithm) The interpolated retrieval type exists to reduce the number of points returned to the display element. For example, Two logs exist, a 1 second 1 hour log and a 15 minute 1 day log. The display can only receive up to 500 numeric points at a time. The user wants 30 minuter of data from the last half hour. If a RAW request was used, the 15m log would return two points and the 1s log would return 1800 points. 2 points fit in the display, but no curve detail is present. 1800 points do not fit in the display and only 5/18 of the data is available. An INTERPOLATED/AVG request on the 1s second log would solve this problem by doing a time average of the 1800 points and reducing the number of points to 500 (or initialValueEntries) if the initalValue Entries was 30, blocks of 60 points would be averaged. This would reduce the points returned from 1800 to 30. **DISPLAY**: The display retrieval type gives the logManager the power to decide whether to use RAW or INTERPOLATED/AVG to retrieve data from a given Numeric log. If the objectName is a logName, display is changed to RAW it the stored points in the interval are less than or equal to the points requested. If the points in the interval are greater than the points requested, the retrieval type is changed to INTERPOLATED/AVG before the request is forwarded to the log.

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric (Continued)

			If the objectName is an accessName, the logManager gathers a list of all TTD and history logs and sorts them in the order best suited to fill the request. By process of elimination, the log Manager finds the best log to fill the request. Depending on the storage interval of the log which was chosen to receive the request, the retrieval type is changed to RAW or INTERPOLATED/AVG for the same reasons as the logName case. An additional feature of the DISPLAY retrieval type is for PRESENT/END_TIME requests, TTD data will be used or the gap between the last history point returned and there present time and to supply cyclic updates. If no history logs exist for an accessName, but TTD does, the logManager will use TTD to give the user what it can.
interpolationAlg	input	HsInterpAlg - HsInterpAlg	hsAVERAGE = Average hsMAXIMUM = Maximum hsMINIMUM = Minimum hsINSTANTANCEOUS = Instantaneous hsSum = SUM hsSUM_OF_SQRS = Sum of SQARE Roots = standard deviation hsNUM_OF_VALUES = Number of values hsWILDCARD = Wildcard
logEntryOrder	input	HsEntryOrder - HsEntryOrder	hsASCENDING = Sort data in ascending order hsDESCENDING = Sort data in descending order
dataSelector	input	HsDataSelector - HsData Selector	When retrieval selects data from the database, ORIGNAL will retrieve only the original data from the database. (NO MODIFIED VALUES). CURRENT will take the current value for each time in the database.

Table 4-50. Parameters to the Numeric Log / Log Manager command GetNumeric (Continued)

queryStart (calcAlg)	input	HsTimeJust - HsTimeJust	queryStart has been overloaded. Its definition has been changed to be calculation algorithm. A calculation must be specified when retrieving by accessName to choose a MIN log over a AVG log or a SUM over a MAX log.
accessID	output	UIntShort - OmfUShort	Identifier of this access. Used for further references. NOTE: User applications should initialize the accessID parameter to 0 (zero).

Table 4-51. Parameters to the Numeric Log / Log Manager command GetNumeric - HsTimeInterval

Parameter	Direction	OTDL- and C-Format	Value/explanation
unit	input	OmfUSmall	Time interval unit. omfSECONDS = 3, omfMINUTES = 4, omfHOURS = 5. omfDAYS = 6 omfWEEKS = 7
pad	input	OmfUSmall	Not used. NOTE: User application should initialize it to 0 (zero).
value	input	OmfUShort	Time interval value expressed in unit defined by the unit parameter. NOTE: Maximum value is 4096. EXAMPLE: A user application using unit = omfSECONDS, and value = 30, will define a 30 seconds interval.

Table 4-52. Parameters to the Numeric Log command AddNumericValue

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectName	input	HsLogNameType	Name of Numeric log.
handle	input	OmfLong	User defined handle identified.
timeType	input	HsTimeType	Type of time stamp that will be used in the logEntryArray. hsLOCAL = 1; Local time stamp, hsUTC = 2; UTC time stamp. NOTE: A user application should use timeType = hsUTC, and then fill in the utc parameter in the logEntryArray using system time for time stamping.
logEntryArray	input	HsNumericEntryLUT	Open array of max 500 history numeric data entries to be added.

Table 4-53. Parameters to the Numeric Log command ModifyNumericValue

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectName	input	HsLogNameType	Name of Numeric log.
handle	input	OmfLong	User defined handle identified.
timeType	input	HsTimeType	Type of time stamp that will be used in the logEntryArray. hsLOCAL = 1; Local time stamp, hsUTC = 2; UTC time stamp. NOTE: A user application should use timeType = hsUTC, and then fill in the utc parameter in the logEntryArray using system time for time stamping.
logEntryArray	input	HsNumericEntryLUT	Open array of max 500 history numeric data entries to be modified.

Table 4-54. Parameters to the Numeric Log commands Add / ModifyNumericValue - HsNumericEntryLUT

Parameter	Direction	OTDL- and C-Format	Value/explanation

Table 4-54. Parameters to the Numeric Log commands Add / ModifyNumericValue - HsNumericEntryLUT

max_is	input	OmfUShort	Max index (dimension - 1) of the array holding data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 499.
last_is	input	OmfUShort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	input	HsNumericLUT	Array of History numeric data entries.

4.9.5 Numeric Log Names

Each Numeric Log object is given a name by AdvaInform History builder when a numeric log is created during configuration. The Numeric Log object names (log names) are by default typically:

`‘$HS<accessname>-<sequencenumber>-o’`

`where <accessname> is the name of the composite log and <sequencenumber> is 1 - 16`.

The data source name for a mapped Master TTD log is typically:

`‘TTD_<network>_<node>_<lognumber>_<variableindex>’`

where `<network>` is the network decade, e.g. 10, 20 ....

NOTE

The TTD name must exist in SQL View TTDMAP, mapped thrue TTD configuration.

The data source name towards a PS object is ´`<objectname>, <attribute>`´

where `<attribute>` typically is VALUE

4.9.6 SQL Views

.

Table 4-55. NUMLOGVAL attributes

NUMLOGVAL Attribute	Oracle Data Type	Description
NAME	VARCHAR2(40)	Name of the Numeric Log.
TIME	DATE	Time stamp of the numeric data value.
MS	NUMBER(10)	Milliseconds part of the time stamp.
ENTRYVALUE	FLOAT (24)	Numeric data value.
ENTRYSTATUS	NUMBER(10)	Status of the numeric data value.
OBJECTSTATUS	NUMBER(10)	Status of the object.
RET_TYPE	VARCHAR2(14)	Definition of retrieval type. Possible values are RAW or INTERPOLATED. RAW returns any data for the actual time interval and INTERPOLATED returns values calculated with the specified interpolation algorithm. Default is RAW
INTERP_ALG	VARCHAR2(14)	Definition of interpolation algorithm. Possible values are AVG, MAX, MIN or INST. Default is AVG
SELECTION	VARCHAR2(10)	Definition of selection for retrieval. Current will return data exactly as currently stored and original will return the original data. Possible values are CURRENT or ORIGINAL. Default is CURRENT
CALC_ALG	VARCHAR2(14)	Definition of calculation algorithm, note that this attribute is only relevant together with seamless retrieval. For mote information about criteria for selection for seamless retrieval refer to AdvaInform History User´s Guide. Possible values are AVG, MAX, MIN, ISNT; SUM, SUM_OF_SQRS, STANDARD_DEV, NUM_OF_VALUES or WILDCARD. Default is WILDCARD
INTERVAL (1)	NUBER(20)	Definition of interpolation interval in seconds. Note that you can specify the interval as a product, i.e. 2*60*60 will result in a time interval of two hours.

(1) If you retrieve interpolated values you have to specify TIME <xxx, TIME> yyy and INTERVAL in the where clause.

The Client is able to fetch, update is not possible, numeric data from a uniquely specified Numeric Log instance MY_LOG like this:

```
SELECT * FROM NUMLOGVAL WHERE NAME=’$HSMY_LOG,VALUE-1-o’
```

The Client can also fetch interpolated values with max as interpolation algorithm, two minutes as interval and a time span of two hours from a Numeric Log like this:

```
SELECT * FROM NUMLOGVAL WHERE NAME=’$HSMY_LOG,VALUE-1-o’ AND
RET_TYPE=´INTERPOLATED´ AND INTERP_ALG=‘MAX´ AND
INTERVAL=2*60 AND TIME >TO_DATE ( ´960301 8:00:00´, ‘YYMMDD
HH24:MI:SS‘) AND TIME <TO_DATE ( ‘960301 10:00:00‘, ‘YYMMDD
HH24:MI:SS‘);
```

When you make a seamless retrieval access it is possible to define a calculation algorithm in the select statement like this:

```
SELECT * FROM NUMLOGVAL WHERE NAME=’MY_LOG,VALUE‘ AND
CALC_ALG=‘MAX‘;
```

NOTE

As NUMLOGVAL is a “SQL*Connect” table, you have to avoid statements like

```
SELECT * FROM NUMLOGVAL;
```

This will cause a lot of load for the station and network. All entries from all numeric logs SQL*Connect Client display will be retrieved!

Also avoid ‘sub-select‘ statements to `NUMLOGVAL (WHERE TIME IN (SELECT xxx FROM yyy)`, which will cause a lot of load.

Instead, try to limit the search, e.g. `SELECT ENTRY VALUE FROM NUMLOGVAL WHERE NAME =‘$HS...‘AND TIME > ... AND TIME <..;`

If you often want just one or very few entries out of NUMLOGVAL, try to define the log as short as possible to increase performance. If you anyway want long logs, define Composite logs with two primary logs to the same data source, one with a minimized Log Capacity to be used by NUMLOGVAL very often, and one with a big Log Capacity to be used less frequent.

Using the Log Name is giving better performance than using the Access Name.

NOTE

Numeric log configuration data can be retrieved with the gtNlog procedure package included in AdvaInform SQL*Connect Programming.

4.10 Message Log Object - MLog

4.10.1 How to Use

Each Message Log object defines one message log within AdvaInform History.
Through the Message Log, History message data can be updated from a User API application.

Message Logs are created and named through AdvaInform History builder functions.

The AdvaInform SQL*Connect - Client makes the AdvaInform History Message Logs appear as a relational database with read access. The database table name used for the Alarm & Event Message Log is hs_msg...

Table 4-56. HS_MESSAGES attributes

HS_MESSAGES Attribute	Oracle Data Type	Description
LOG_NAME	CHAR(15)	Name of the Message Log.
OBJECT_NAME	CHAR(13)	Name of the object that generated the message.
EVENT	CHAR(20)	The event as text.
REASON	CHAR(20)	The event reason as text.
LOCAL_TIME	CHAR(84)	The time stamp in local time as text.

The Client is able to fetch data from the Alarm & Event Message Log for a user specified object instance MY_AIOBJ, like this:

```
SELECT * FROM HS_MESSAGES
    WHERE LOG_NAME=’$HSEAMSGLOG-1-o’
    AND OBJECT_NAME=’MY_AIOBJ’;
```

For a complete description of the SQL access of the Alarm & Event Message Log please refer to the *AdvaInform Basic Functions User’s Guide*.

4.10.2 Overview

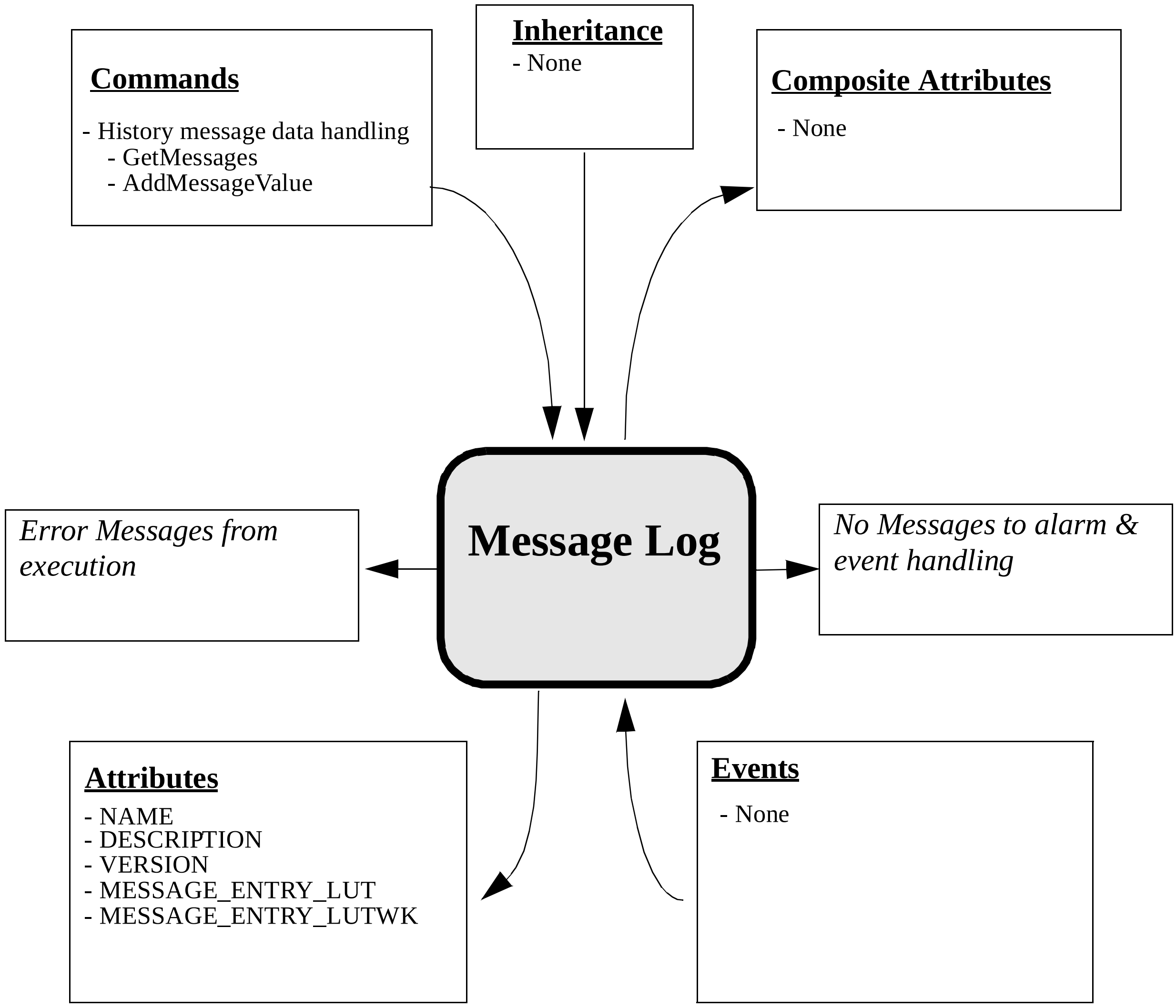


Figure 4-11. Overview of the Message Log

4.10.3 Attributes

Table 4-57. Message Log object type attributes

Attribute	OTDL - C Data Type	Description
NAME	OmfChar(740)	Name of Message Log.
DESCRIPTION	OmfChar(41)	Description of Message Log.
VERSION	OmfChar(21)	Software Release Version of Message Log.
MESSAGE_ ENTRY_LUT	HsMessage EntryLUT	Open array of max 225 historical messages with Local & UTC time stamp. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`
MESSAGE_ ENTRY_LUTWK	HsMessage EntryLUTWk	Open array of max 125 historical messages with Local & UTC time stamp and all different parts of the messages data as separate attributes. The definition is found in the file `/opt/advant/TypeDir/include/omfAttributeT.h`

Table 4-58. Message Log HsMessageEntryLUT & HsMessageLUTWk Open Array type attributes

Attribute	OTDL - C Data Type	Description
max_is	OmfUShort	Max index (dimension - 1) of the array holding data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 224. for LUT and 124 for LUTWk
last_is	OmfUShort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	HsMessageLUT or HsMessageLUTWk	Array of History message data entries.

Table 4-59. Message Log HsMessageLUT & HsMessageLUTWk type attributes

Attribute	OTDL - C Data Type	Description
localTime	HsTimeStamp - BciTimeStamp	Time stamp of this message value entry in Local time format.
utc	HsTimeStamp - BciTimeStamp	Time stamp of this message value entry in UTC time format.
historyMsgID	OmfULong	Identifier of this message. Used for further references.
msgCount	OmfUSmall	Not used.
entryStatus	HsLogStatus Type	Status of this message data entry. For a description of the HsLogStatusType type please refer to the HsLogStatusType bits table in the Numeric Log section. See Table 4-48
message	OmfChar(255)	This message text.
procObjName	OmfChar(33)	Name of process object generating the message. NOTE: the procObjName is only present in HsMessageLUTWk
sectionName	OmfChar(33)	The section the process object belongs to. NOTE: the sectionName is only present in HsMessageLUTWk
batchID	OmfChar(41)	The batch identity. Not used. NOTE: the batchID is only present in HsMessageLUTWk
unitID	OmfChar(13)	The unit identity. Not used. NOTE: the unitID is only present in HsMessageLUTWk
sequenceID	OmfChar(13)	The sequence identity. Not used. NOTE: the sequenceID is only present in HsMessageLUTWk
eventReason	OmfULong	The reason for the event generating the message. NOTE: the eventReason is only present in HsMessageLUTWk
eventProperty	OmfULong	The property involved in the event generating the message. NOTE: the eventProperty is only present in HsMessageLUTWk
netwID	OmfULong	The identity of the network where the message was generated. NOTE: the netwID is only present in HsMessageLUTWk
nodeID	OmfULong	The identity of the node where the message was generated. NOTE: the nodeID is only present in HsMessageLUTWk

Table 4-59. Message Log HsMessageLUT & HsMessageLUTWk type attributes (Continued)

Attribute	OTDL - C Data Type	Description
priority	OmfULong	The priority of the message. NOTE: the priority is only present in HsMessageLUTWk
msgType	HsMessageType	The type of the message. hsALARM = 1, hsBILLBOARD = 2, hsPARAMETER_CHANGE = 3, hsLADDER_LOGIC = 4, hsSPC = 5, hsALARM_ACK = 6, hsDIAGNOSTIC = 7, hsBATCH_MSG = 8. NOTE: the msgType is only present in HsMessageLUTWk

4.10.4 Commands

The Message Log has the following commands:

GetMessages

With **GetMessages** it is possible to retrieve up to 225 for LUT and 125 for LUTWk historical messages from a History Message Log. The request command must be combined with a subscription for one of the attributes MESSAGE_ENTRY_LUT or MESSAGE_ENTRY_LUTWK.

Syntax

```
GetMessages ( objectName, selectionTime, timeType,
              timeJustification, timeInterv, historyMsgID,
              messageType, retrieveKeyType, retrieveKey,
              initialMsgEntries, initalEventEntries,
              cyclicMsgEntries, cyclicEventEntries,
              logEntryOrder, directionOfRequest, queryStart,
              language )
```

AddMessageValue

With **AddMessageValue** it is possible to update up to 125 historical message data values in a History Message Log.

Syntax

```
AddMessageValue ( timeType, logEntryArray )
```

NOTE:

To update History message data in the Message Log using a User API application, the **bciExecOperation()** User API function should be used with the following parameters/data:

- The command/operation **AddMessageValue**.
- Include files **omfTypeDirectory.h** and **omf_MLog.h**.

In the following a description of the AddMessageValue parameters are given.

Table 4-60. Parameters to the Message Log / Log Manager command GetMessages

Parameter	Direction	OTDL- and C-Format	Value/explanation
objectName	input	HsLogNameType - OmfChar(740)	Name of Message Log to collect data from.
selectionTime	input	HsTimeStamp - BciTimeStamp	Time reference point for the message data to collect. Given as a time stamp value.
timeType	input	HsTimeType	The type of the selectionTime time stamp. hsPRESENT = 0; Current time, hsLOCAL = 1; Local time stamp, hsUTC = 2; UTC time stamp. NOTE: If hsPRESENT, current time will be the reference time, and the selectionTime is not used.
timeJustification	input	HsTimeJust	Justification if the selectionTime indicates the start time or end time of the requested interval. hsSTART_TIME = 0; From time, hsEND_TIME = 1; Up to time. EXAMPLE: A user application using timeJustification = hsEND_TIME & timeType = hsPRESENT, will retrieve historical data up to current time (youngest data available).
timeInterv	input	HsTimeInterval	The time interval of the message data collection range. Specified as a unit and a value.
historyMsgID	input	OmfULong	Reference identity given for multiple accesses. NOTE: User applications should initialize the historyMsgID parameter to 0 (zero) at first.

Table 4-60. Parameters to the Message Log / Log Manager command GetMessages (Continued)

messageType	input	HsMessageType	Key for type of messages to retrieve. hsUNTYPED = 0; All types, hsALARM = 1, hsBILLBOARD = 2, hsPARAMETER_CHANGE = 3, hsLADDER_LOGIC = 4, hsSPC = 5, hsALARM_ACK = 6, hsDIAGNOSTIC = 7, hsBATCH_MSG = 8. NOTE: User Applications should normally use the hsUNTYPED or hsALARM message type key.
retrieveKeyType	input	HsMsgKeyType	Type of key for message retrieval filtering within messageType. hsALL_MESSAGES = 0; No filter, hsPROCESS_OBJECT = 1, hsSECTION_NAME = 2, hsBATCH_ID = 3, hsUNIT_ID = 4, hsSEQUENCE_ID = 5, hsEVENT_PROPERTY = 6, hsNETWORK_ID = 7, hsNODE_ID = 8, hsMSG_PRIORITY = 9. NOTE: User Applications should normally use the hsALL_MESSAGES or hsPROCESS_OBJECT message filtering key.
retrieveKey	input	HsMsgKeyValue	Union with the value of type retrieveKeyType to match.
retrieveKey.unionTag		OmfUShort	0 = String value, 1 = Integer value.
retrieveKey.u		a union struct of:	Depending on retrieveKeyType:
ret--Key.u.stringKey OR ret--Key.u.longIntKey		OmfChar(41) OR OmfULong	A String value OR An Integer value.
initialMsgEntries	input	OmfUShort	Number of message data entries to retrieve in initial request. NOTE: maximum is 225.

Table 4-60. Parameters to the Message Log / Log Manager command GetMessages (Continued)

initialEventEntries	input	OmfUShort	Not used. NOTE: User application should initialize to 0 (zero)
cyclicMsgEntries	input	OmfUShort	Number of message data entries to retrieve in each cycle.
cyclicEventEntries	input	OmfUShort	Not used. NOTE: User application should initialize to 0 (zero)
logEntryOrder	input	HsEntryOrder	Time stamp order of the message data retrieved. hsASCENDING = 0; Normal, hsDESCENDING = 1; Backwards. NOTE: User applications should use hsASCENDING, which retrieves message data exactly as stored in timed order.
directionOfRequest	input	HsRequestDir	Direction of retrieval request. hsFORWARD = 0; Normal, hsBACKWARD = 1; Backwards. NOTE: User applications should use hsFORWARD, which retrieves message data exactly as stored.
queryStart	input	HsTimeJust	Not used. hsSTART_TIME = 0; Normal, hsEND_TIME = 1; Backwards. NOTE: User applications should use hsSTART_TIME, which retrieves message data exactly as stored.
language	input	HsNameType	Not used. NOTE: User application should initialize to an empty string.

Table 4-61. Parameters to the Message Log / Log Manager command GetMessages - HsTimeInterval

Parameter	Direction	OTDL- and C-Format	Value/explanation
unit	input	OmfUSmall	Time interval unit. omfSECONDS = 3, omfMINUTES = 4, omfHOURS = 5. omfDAYS = 6 omfWEEKS = 7
pad	input	OmfUSmall	Not used. NOTE: User application should initialize it to 0 (zero).
value	input	OmfUShort	Time interval value expressed in unit defined by the unit parameter. NOTE: Maximum value is 4096. EXAMPLE: A user application using unit = omfHOURS, and value = 24, will define a day long interval.

Table 4-62. Parameters to the Message Log command AddMessageValue

Parameter	Direction	OTDL- and C-Format	Value/explanation
timeType	input	HsTimeType	Type of time stamp that will be used in the logEntryArray. hsLOCAL = 1; Local time stamp, hsUTC = 2; UTC time stamp. NOTE: A user application should use timeType = hsUTC, and then fill in the utc parameter in the logEntryArray using system time for time stamping.
logEntryArray	input	HsMessageEntryLUTWk	Array of History message data entries to add.

Table 4-63. Parameters to the Message Log command AddMessageValue - HsMessageEntryLUTWk

Parameter	Direction	OTDL- and C-Format	Value/explanation
max_is	input	OmfUShort	Max index (dimension - 1) of the array holding message data. NOTE: User application should initialize max_is before requesting data. Maximum max_is is 125.
last_is	input/output	OmfUshort	Last index occupied. NOTE: User application should Initialize last_is to omfOPEN_ARRAY_IS_EMPTY before requesting data, and check value against omfOPEN_ARRAY_IS_EMPTY before accessing array contents.
array	output	HsMessageLUTWk	The array of History message data entries.

Table 4-64. Parameters to the Message Log command AddMessageValue - HsMessageLUTWk

Parameter	Direction	OTDL- and C-Format	Value/explanation
localTime	input	HsTimeStamp - BciTimeStamp	Time stamp of this message value entry in Local time format. The localTime parameter is used if timeType is set to hsLOCAL. NOTE: Should normally not be used by user applications.
utc	input	HsTimeStamp - BciTimeStamp	Time stamp of this message value entry in UTC time format. NOTE: A user application should use timeType = hsUTC, and then fill in the utc parameter in the logEntryArray using system time for time stamping.
historyMsgID	input	OmfULong	History Identifier of this message. Used for further references. NOTE: Should be initialized to 0 (zero).

Table 4-64. Parameters to the Message Log command AddMessageValue - HsMessageLUTWk (Continued)

msgCount	input	OmfUSmall	Not used. NOTE: Should be initialized to 0 (zero).
entryStatus	input	HsLogStatusType	Status of this message data entry. For a description of the HsLogStatusType type please refer to the HsLogStatusType bits table in the Numeric Log section. NOTE: Should be initialized to 0 (zero), no status bits set.
message	input	OmfChar(255)	Message string data entry. NOTE: Should be initialized to empty string.
procObjName	input	OmfChar(33)	Name of process object generating the message.
sectionName	input	OmfChar(33)	The process section the process object belongs to.
batchID	input	OmfChar(41)	The batch identity. Not used. NOTE: Should be initialized to empty string.
unitID	input	OmfChar(13)	The unit identity. Not used. NOTE: Should be initialized to empty string.
sequenceID	input	OmfChar(13)	The sequence identity. Not used. NOTE: Should be initialized to empty string.
eventReason	input	OmfULong	The reason for the event generating the message.
eventProperty	input	OmfULong	The attribute involved in the event generating the message.
netwID	input	OmfULong	The identity of the network where the message was generated.
nodeID	input	OmfULong	The identity of the node where the message was generated.

Table 4-64. Parameters to the Message Log command AddMessageValue - HsMessageLUTWk (Continued)

priority	input	OmfULong	The priority of the message.
msgType	input	HsMessageType	The type of the message. hsALARM = 1, hsBILLBOARD = 2, hsPARAMETER_CHANGE = 3, hsLADDER_LOGIC = 4, hsSPC = 5, hsALARM_ACK = 6, hsDIAGNOSTIC = 7, hsBATCH_MSG = 8. NOTE: User Applications should initialize it to 0 (zero) message type.

4.10.5 Message Log Names

Each Message Log object is given a name by AdvaInform History builder when a message log is created during configuration. The Message Log object names (log names) are by default typically:

`‘$HS<accessname>-1-o’.`

```
   where <accessname> is composite log name.
```

The IMS Alarm & Event Message Log object name is:

`‘$HSEAMSGLOG-1-o’.`

4.11 RTA System Object

4.11.1 How to Use

The RTA system object type provides functionality to retrieve the CPU load and memory usage of the RTA board.

4.11.2 Overview

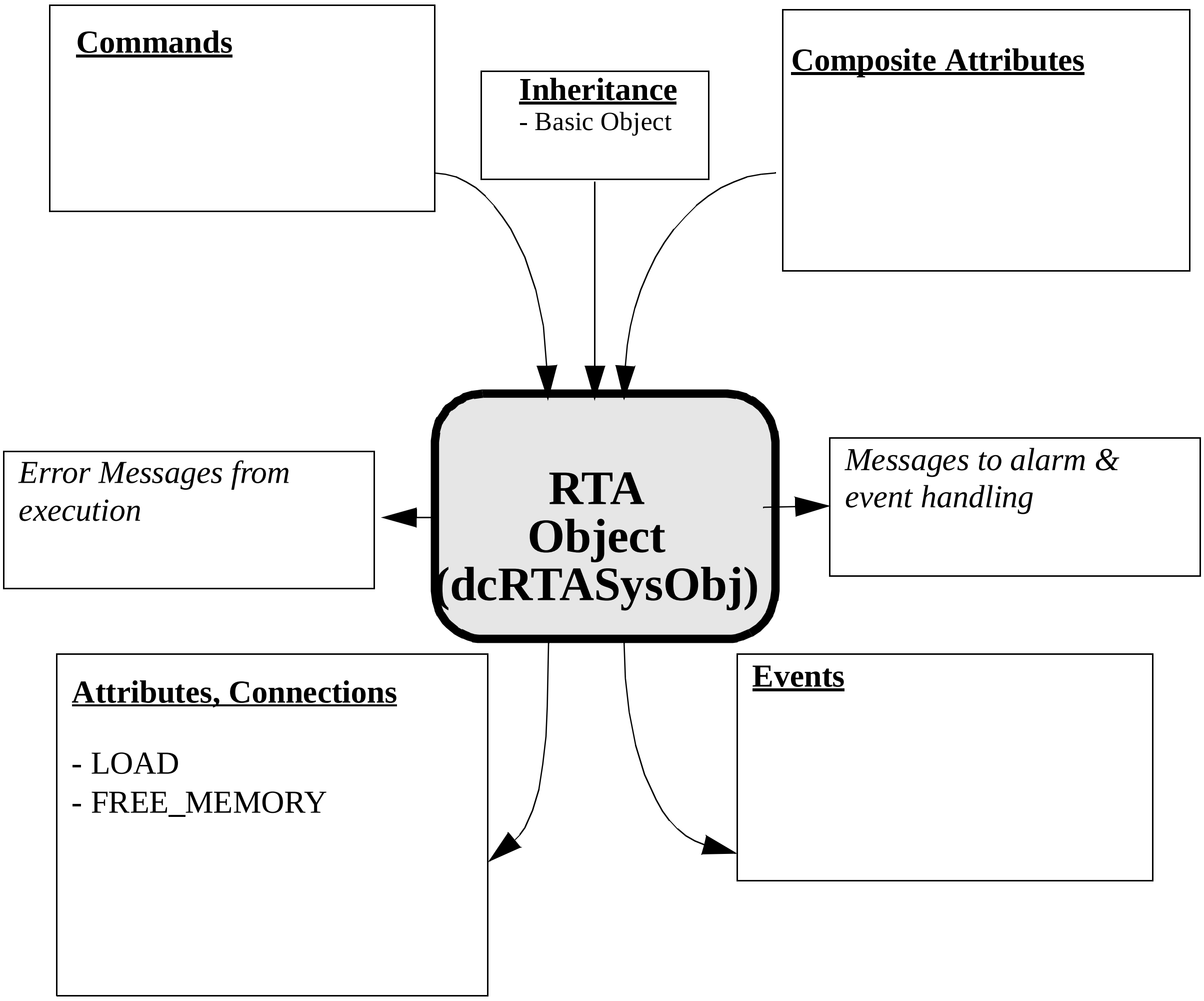


Figure 4-12. Overview of the RTA System Object Type

4.11.3 Attributes

Table 4-65. RTA system object type attributes

Attribute	OTDL - C Data Type	Description
LOAD	OmfSmall	RTA CPU load
FREE_MEMORY	OmfLong	RTA memory usage

4.11.4 Connections

The RTA System object has no connections.

4.11.5 Composite Attributes

The RTA system object has no composite attributes.

4.11.6 Commands

The RTA system object has no commands

4.11.7 Event

The RTA system object has no events.

3BSE 004 008R0301

September 1997